1971

An information theory approach to a fault location problem

Campbell, David Russell

AN INFORMATION THEORY APPROACH TO A FAULT LOCATION PROBLEM

DAVID RUSSELL CAMPBELL

United States
Naval Postgraduate School

THESIS

AN INFORMATION THEORY APPROACH

TO A FAULT LOCATION PROBLEM

by

David Russell Campbell

Thesis Advisor: R. W. Butterworth

September 1971

An Information Theory Approach to a Fault Location Problem

by

David Russell Campbell
Lieutenant, United States Navy
B.S., University of New Mexico, 1963

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1971

ABSTRACT

The fault location model under investigation consists of an n-component series system known to have exactly one failed component. Component positions in the system are taken as fixed. A component is either working or failed. Components work or fail independently of each other, with their a priori reliabilities taken as given but not necessarily equal. Group testing to locate the failed component is sequential, binary and dichotomous in nature with certain results. The only costs are the number of tests made. The three solution procedures investigated are (1) a dynamic programming formulation, (2) a sequential halving procedure, and (3) a procedure based on information theory. The criteria for optimality are minimization of the expected number of tests required and minimization of the maximum number of tests required.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. THE GENERAL FAULT LOCATION PROBLEM

A. PROBLEM STATEMENT

The most general form of the fault location problem may be stated as follows: One is given a population consisting of individual elements. This set is partitioned into two disjoint and exhaustive subsets. The problem posed is to specify procedures designed to locate in an efficient manner those elements belonging to the first subset.

The property which distinguishes those elements of the first subset is connected with some sort of deficiency or fault in most applications, hence the term "fault location problem".

B. DISTINGUISHING MODEL CHARACTERISTICS

The general fault location problem encompasses many specific models.

1. Population Size

Model population size is either finite or countably infinite. The assumption of an infinite population can simplify calculations in many instances.

2. Number of Defectives Known to Exist

"Number of defectives" here is equivalent to the cardinality of the first subset in the general formulation. This number is generally taken at the outset of the problem to be either completely unknown, one or more, exactly one, or unknown but greater than some specified number.

3. Probability of Elements Being Defective

The *a priori* probability of being defective is given for each element. With no information as to the number of existing defectives

each element is defective with this probability, independent of the state of other elements. Information on the number of existing defectives yields a posterior probability of being defective for each element given this information. These posterior probabilities are not independent.

4. Test Outcomes

A test is an action taken to identify elements belonging to the defective set. Tests may result in binary outcomes, i.e., pass/fail where these terms are appropriately defined; or they may result in multiple outcomes.

5. Certainty of Test Results

Test results may be certain, i.e., indicate the true state of nature; or they may be uncertain, in which case probabilities of erroneous test outcomes are specified.

6. Number of Elements Included in a Single Test

The number of elements included in a test may be one or more, the latter termed group testing.

7. Test Costs

Test costs are either assumed to be equal for all tests, or unequal and specified. Costs are measured in terms of the appropriate scarce resource, e.g., time, money, etc.

II. OVERVIEW OF WORK IN FAULT LOCATION PROBLEMS

The following discussion of work which has been done in the general area of fault location problems is not claimed to be either an exhaustive enumeration of all such work or a summary of all the results achieved in the papers mentioned. Rather, it is intended to give the reader an appreciation for the scope of the general problem, a brief description of some of the work which has been done, and an idea of just where the model of this thesis fits.

The first work in this area is generally attributed to Dorfman [7] in 1943. The problem he considered was involved with the requirement levied by World War II on the Public Health Service and the Selective Service System to weed out syphilitic men from the large number of incoming inductees. Blood samples were tested using the "Wasserman-type" blood test which detects the presence of a syphilitic antigen. Prior to Dorfman's efforts, blood samples were tensted on an individual basis, requiring a large effort on the part of laboratory personnel. Dorfman found that one could pool the blood samples from a number of men and test this pooled sample. If none of the men was infected, the entire group could thus be passed with a single test. The blood sample drawn from each man was sufficient to conduct two tests. The remaining blood for each member of an infected group could thus be tested individually to identify the infected member(s). Dorfman considered the population of inductees to be infinite. He established optimum group sizes and relative expected test costs (as opposed to individual sample testing) for selected a priori probabilities that any member of the population would be infected. This became known as the "blood testing problem". Dorfman's work showed the advantage of group testing to be inversely

related to the a priori probability of infection, with the break-even point occurring at a probability of about 0.15. He considered the tests to be binary in nature with certain results.

In 1960, Ungar [25] expanded on Dorfman and others' work on the blood testing problem. He considered the population to be of finite size n, each member with the a priori probability p of being infected. His results show that the range of p for which group testing is advantageous is $0 \leq p < 1/2(3 - \sqrt{5}) \doteq 0.38$. Ungar's group testing allows grouping of individuals for all tests.

Finucan [8] in 1964 also extended the results of Dorfman in an algebraic treatment for Dorfman's two-stage method (group testing at the first stage, then individual testing at the second), and also for methods using three or more stages. As in Dorfman's work, Finucan assumes the population to be infinite with each member having an equal prior probability of being infected. As in Dorfman and Ungar, minimization of the expected number of tests required is the criterion adopted for optimality.

Sobel and Groll [21] in 1959 also extended Dorfman's original efforts. In their model the population size is finite, the number of elements in each group test is not necessarily constant, and if a group test fails, each element is not necessarily tested individually. Several procedures are developed with conditions for their optimality under ranges of p values. The criterion used is the expected number of tests required to eliminate all defectives. Several industrial applications are also cited. This work was extended by Sobel [20,23] in 1960 and 1964. Kumar and Sobel [24] considered a variant of this model in 1970. They begin with an infinite population, each of whose elements has an equal prior probability of being defective. In this model an optimum group testing

procedure is obtained in the sense that it minimizes the expected number of tests required to find the first defective.

Black [1] in 1965 formulated a search model in which the finite population consists of "regions" in which a single target may lie. A test consists of a "look" in a region for the target. The binary results of these tests are uncertain in that each region is assigned m_i, a conditional miss probability, i.e., probability that given the target is in the i^{th} region, it will not be detected on a single look there. Each region is also assigned a prior probability that the target is there (p_i), and a cost per look (c_i). His solution requires the computation of a figure of merit, $p_i(1-m_i)m_i^{n-1}/c_i$, for each of the i regions and for as many of n looks as desired. These values are placed in a two-dimensional array (by n and i). The order of search of the regions is in decreasing order of their figures of merit. Black also derives a means for calculation of the expected cost of the search. His model is equivalent to individual element testing with the cost of searching each region independent of the search effort previously expended in other regions. He references Matula [16], a student of Blackwell, who originally drew upon the results achieved by Blackwell [2] in his "ball-in-box" problem. Klein [5] in 1967 also formulated a search model involving a single target to be located, which employs Markovian decision models.

Gaddess [6] addressed in 1969 a related problem for which very few results have been obtained. His work concerns the specification of test points within a modular combinational logic circuit with the objective of insuring diagnosability to the module level of the overall circuit, i.e., insuring that there exists at least one sequence of binary test

inputs which distinguishes any two single faults in separate modules. He derives three techniques for allocating test points: (1) A method analogous to the prime implicant covering problem of classical switching theory, (2) a graphical worst case analysis, and (3) a combination of the first two methods.

Zimmerman [26] in 1959 investigated a model in which the population is finite with exactly one defective element. Each element has a known (but not necessarily equal) probability of being defective. Testing is binary with certain results. Each test partitions the elements into two groups; the test outcome reveals which group contains the defective element. This group then comprises the elements for the next test. Testing is complete when the defective is found. Optimality is judged on the expected number of tests required. Zimmerman views the problem as a repeated combination of the elements as opposed to repeated division. His results involve the combination at any stage of the two elements having the smallest prior probabilities. These results appear to the author to be equivalent to a coding procedure developed by Huffman [12] in 1952.

Sandelius [19] in 1961 examined a particular case of the model previously treated by Zimmerman. Sandelius' model assumes that all elements have an equal prior probability of being defective. Testing is as described above for Zimmerman. To state Sandelius' theorem, we define the following:

$$n = m_o = 2^{k_o} + r_o \ ,$$

where n = number of elements to be tested

m_i = number of elements remaining to be tested after i tests have been conducted

$$k_i \in \{0,1,2,\ldots\}$$

$$r_i \in \{0,1,2,\ldots,(2^{k_i} - 1)\}$$

Sandelius proves the following theorem: When all p_i are equal the minimum expected number of tests is $k_0 + (2r_0/n)$, and this will be attained if and only if the number of elements of each of the two groups which are tested in the $(h + 1)^{th}$ test ($h = 0,1,2,\ldots$) belongs to the closed interval $[2^{k_h-1}, 2^{k_h}]$.

Johnson [18] in 1956 couched his models in terms of electronics equipment to be checked out. His population is finite and considered from two related levels. In the first, the population is composed of the electronic components which form the system. The second level population consists of the individual parts of which each component is composed. Sequential binary testing with certain results takes place on the components level first, then on the parts level of a located failed component. On each level elements are tested individually. Components and their associated parts are assigned prior probabilities of failure, and required test times (broken down further into the sum of the time for removal for testing, the time for the actual test, and the time for replacement). Optimum test sequences minimize the expected delay time, measured from the time the system is initially tested to the time it works. Johnson considers both multiple failures in the system, and the case of one failure.

Gluss [10] in 1959 considered a model equivalent to Johnson's case of exactly one failure described above, with the exception that test results are no longer certain. Attached to each element is a conditional probability that a single test of that element will fail to indicate

failure, given the element has actually failed. Firstman and Gluss [9] in 1960 expanded Gluss' model to include "false alarms", i.e., tests which indicate failure when no failure has occurred.

Butterworth [4] expanded the results of Johnson to examine specific types of systems. His models treat optimum sequential binary testing with certain results of individual components from a finite population. Each component's prior probability of failure and the time required to test it are known. The systems treated are the k-out-of-n type (k/n), i.e., the system works if and only if k or more of its components work. The series (n/n) and parallel (1/n) systems are included as special cases. In his first model a feasible testing procedure must determine the system state, assuming the components to be independent. In other models for which it is assumed that the system has failed, a feasible testing procedure must locate all failed components. Testing procedures are judged in terms of the expected time required to complete testing.

Brulé, Johnson and Kletsky [3] in 1960 investigated models phrased in the electronics system form. The individual components compose the finite population. They consider first a model allowing multiple component failures. Their goal is to construct a general testing diagram to show in node and arc form the successive system states, as represented by binary n-tuples, and the sequential tests to perform in locating the failed components. They conclude that even simple systems can require extremely complex testing diagrams, but that this requirement may be reduced to tractable proportions if the allowed number of failed components is reduced to exactly one. Their resulting testing diagram constructions are then based on this stipulation. They also examine

sequential binary test procedures for two special cases. For both cases, testing is performed as described above for Zimmerman. The first special case, is termed the equal cost-equal probability case, in which the prior probabilities of failure and test costs for all components are equal. They cite a procedure termed the "half-split technique" which is equivalent to that described above in Sandelius' work. In this case, this procedure leads to an optimum solution in terms of minimizing the exptected cost of testing. For this special case, they claim that this procedure is a minimax solution, i.e., it minimizes the maximum possible cost. They further claim that it yields the minimax solution when the probabilities of failure are unequal. The author refutes both these claims. A counterexample is attached at the end of Appendix C. Their second special case is the equal cost-unequal probability case. They show an analogy between this situation and that described by Huffman [12] in his optimum coding problem. To employ Huffman's procedure the components must be arrangeable in order of decreasing reliability, thus one must be able to group them in this manner under the testing procedure.

Kletsky [15] in 1960 expanded on both the work mentioned above by Brulé, Johnson and himself and that of Johnson [13]. He shows an example of the application of a figure of merit based upon information theory in the formulation of a testing diagram corresponding to an "efficient" sequential test procedure. The example system is the power supply of a standard USAF communications receiver. Estimation of posterior probabilities of failure given that the equipment has failed for each component is shown through a frequency analysis of the equipment failure history. Implicit in these calculations is the assumption that any equipment failure results from exactly one failed component. The figure of merit

F_k representing the ratio of ambiguity removed by the k^{th} test to the cost of performing the test, is as follows:

$$F_k = \frac{-P\log_2 P - (1-P)\log_2(1-P)}{C_k} ,$$

where P is the probability that the test will pass and C_k is the cost of performing the test. Kletsky's procedure is as follows:

1) Evaluate F_k for each of the possible tests.
2) Choose the test with the largest F_k.
3) Alter the cost of performing the remaining tests on the basis of having performed this and other previous tests.
4) Alter the probability of passing for each of the remaining tests on the basis of knowledge gained by having performed this and other previous tests.
5) Repeat the procedure until the entire sequential testing diagram is determined.

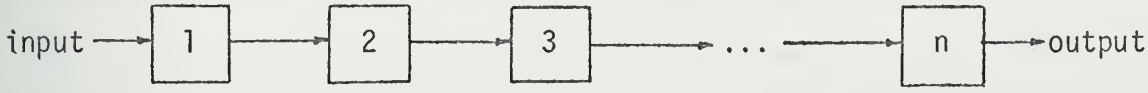
As an indication of the "efficiency" of the above-described procedure, Kletsky offers only the statement that "...the information theory technique yields a diagnostic procedure which is not essentially different from that which would be used by a competent technician".

Kovacs [11] in 1968 expanded on Kletsky's work. He derives an equivalent figure of merit for multiple outcome tests which reduces to the above form for binary tests. He also discusses the use of multiple outcome versus binary tests, and presents a method for the incorporation of multiple outcome tests on a binary test diagram. In evaluation of the information theory figure of merit approach, Kovacs states simply, "...this approach opens the door to...the most rapid and efficient checkout procedure."

III. MODEL DESCRIPTION

A. SYSTEM

The population under investigation in this model can be described as a series system of a finite number of components. In terms of the system descriptions used by Butterworth [4], it is an n/n system in which the system works if and only if all its components work. If the system does not work, it is said to be failed. The system is most easily visualized as an electronic system with a single signal path, as shown in Figure 1.



An n-Component Series System
Figure 1

B. COMPONENT ATTRIBUTES

Component positions in the system are taken as fixed, such as in a series resistor network. As a convention, the system is as shown in Figure 1, with the input to the left and the components numbered sequentially from left to right.

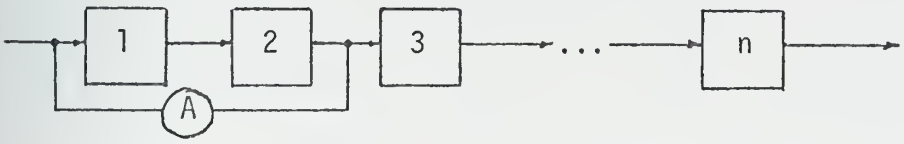
A component may be in one of two states, i.e., working or failed. Components work or fail independently of each other. The a priori probability, p_i^o for i = 1,2,...,n, that a component is working is taken as given, but not necessarily equal for all components. This probability

is sometimes called the reliability of the component, and may be expressed as the probability that the component will perform satisfactorily throughout the duration of some specified mission. The determination of these probabilities is not considered here.

The system is assumed initially to have exactly one failed component.

C. TESTING

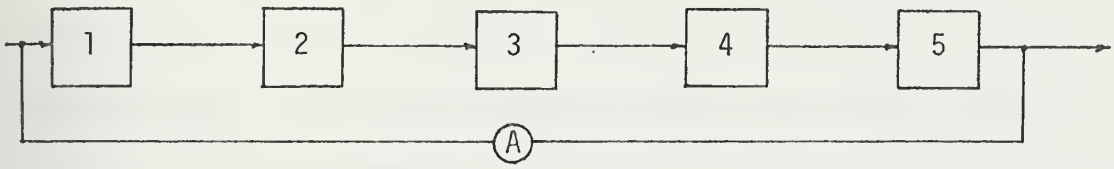
Testing is to be sequential, binary, and dichotomous in nature with certain results. It is dichotomous in that each test partitions the set of components known to contain the failed component into two subsets. It is sequential in that the outcome of any test will determine the next test to be conducted, thus the result of each test is evaluated prior to conducting the next test. Testing is binary in that only pass/fail results are obtained. The only costs are the number of tests made. We may think of the tests being conducted as testing across a number of resistors for continuity using an ammeter with one probe of the ammeter fixed at the left end of the system. A test across the first two components is shown in Figure 2. If this test "passes", it indicates continuity through components 1 and 2, indicating that the failed component lies to the right of component 2. If the test "fails", it indicates no continuity through the first two components, thus one of them must be failed (since test results indicate the true state of nature). "Failed" in this context is equivalent to "open" in the electrical sense.



Testing Across Components 1 and 2

Figure 2

Suppose the test in Figure 2 passes. Then the next test might be say as shown in Figure 3, where the set of components 3 through n is partitioned into sets consisting of components 3 through 5, and 6 through n. If this test fails, the failed component is now known to be either component 3, 4, or 5; and if it passes, the failed component is now known to lie to the right of component 5.



Testing Across Components 1-5
Figure 3

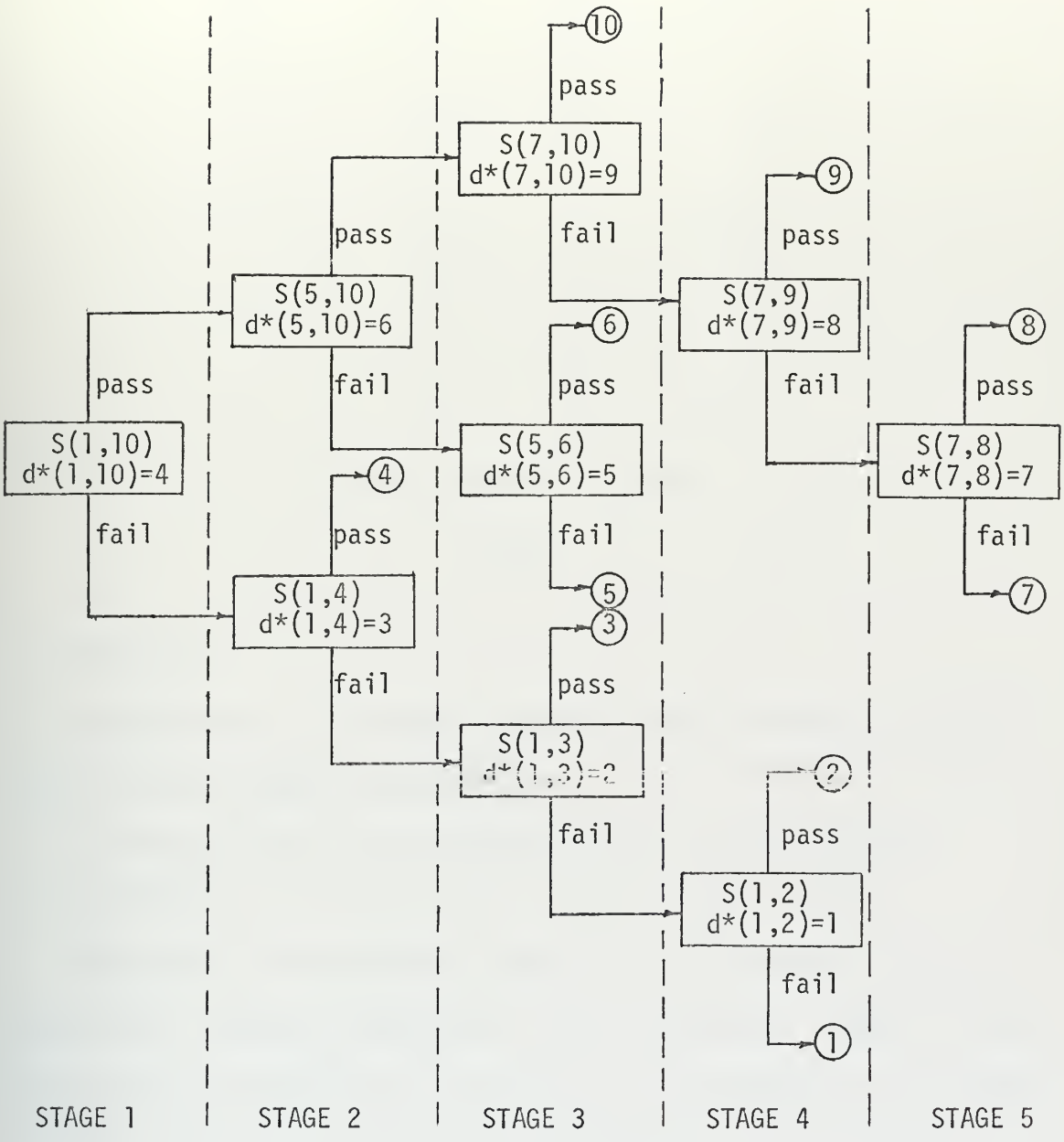
This sequential testing scheme requires a decision of where to place the right hand probe for each succeeding test, based upon the results of all previous tests. To specify any particular solution, these decisions must be specified for both pass and fail results for each test. The number of tests required to locate the failed component will be a random variable, depending on the test plan adopted and the component which has actually failed. We will let N represent this integer-valued random variable, and denote its expected value by $E(N)$.

Utilization of our testing procedure will result at any point in the set known to contain the failed component being composed only of components adjacent to each other, such as components i through j, $i \leq j$. We shall use this fact to describe the system's state at any point in the testing procedure as follows: Let $S(i,j)$ indicate that components i through j inclusive comprise the set known to contain the failed component. Then let $d(i,j) = k$ indicate the decision to test across

the first k components when the system state $S(i,j)$ is realized. Note that $k \in \{i,i+1,\ldots,j-1\}$. Further, if this test passes, we have $S(k+1,j)$, and if it fails $S(i,k)$. Testing is complete when we reach $S(i,i)$ for some i, for then component i is known to be failed.

Diagrammatic representation of a derived test plan solution is best represented in the node-arc form as shown in Figure 4. In this representation, the rectangular nodes show sequential system states which may result throughout the procedure and their associated decision variable values. The circular nodes indicate the located failed component. This representation shows all possible paths through the procedure, each terminating with the location of the failed component. For example, suppose component three is actually the failed component. The test across the first four components fails, resulting in state $S(1,4)$. The next test across the first three components also fails, resulting in state $S(1,3)$. The third test across the first two components passes, indicating component three to be failed (the terminal state $S(3,3)$). Thus, three tests are required under this procedure to locate the third component.

For purposes of compactness, we wish to reduce the diagram of Figure 4 to tabular form. An equivalent procedure is depicted in Table I. The "TEST" and "STATE" columns are self-explanatory. The "DEC" column denotes the decision corresponding to the state to its left. The "COMP" column denotes the component(s) which, if failed, would be located by the test. This tabular form will be utilized to describe solutions to example cases.



Sample Test Plan for 10-Component System

Figure 4

TEST	STATE	DEC	COMP
1	1,10	4	
2	1, 4	3	4
	5,10	6	
3	1, 3	2	3
	5, 6	5	5, 6
	7,10	9	10
4	1, 2	1	1, 2
	7, 9	8	9
5	7, 8	7	7, 8

Tabular Form of Test Plan

TABLE I

D. OBJECTIVE

Our objective is to examine several solution procedures based on the following criteria for optimality:

1) Minimization of the expected number of tests required.

2) Minimization of the maximum number of tests required.

Minimization of the expected number of tests required is the criterion most frequently encountered in the literature. Implicit in the use of such an expected value criterion is the assumption that testing is to be repetitious, yielding good results over the "long haul". In many cases, the results of any one realization of a system test may be disastrous in terms of the number of tests actually required. To hedge against this kind of disastrous realization, one may instead employ the criterion of minimizing the maximum number of tests required to locate the failed component. This is termed a "minimax" criterion.

Of course, if one could devise a testing procedure which would be optimum in terms of both criteria, he might then consider the procedure

to be ideal. In general, however, one would expect that tradeoffs would exist between the two. In the practical sense then, one must examine the attributes of alternative solutions in order to choose the one which best suits his needs.

We will examine three procedures. The first procedure involves the solution of a dynamic programming formulation of the problem. This results in a test plan which minimizes the expected number of tests required.

The second procedure, termed the sequential halving procedure, is shown to minimize the maximum number of tests required.

The third procedure, termed the information theory approach, is a heuristic procedure based upon maximizing the expected reduction in the uncertainty of the failed component's location at each succeeding test. This procedure is not claimed to be optimum in terms of either criterion, yet its ease of computation suggests it to be worthy of investigation.

IV. MODEL SOLUTIONS

A. DYNAMIC PROGRAMMING FORMULATION

1. Nomenclature

n = number of components which comprise the system.

q_i = probability that component i is failed, given that the system is failed with exactly one failed component. $\sum_{i=1}^{n} q_i = 1$.

$S(i,j)$ = system state denoting that the failed component is known to be one of the components $i,i+1,\ldots,j-1,j$. Note that $i \leq j$.

$Q_k(i,j)$ = minimum expected number of tests required to locate the failed component beginning in state $S(i,j)$ and testing across the first k components on the next test. Note that $k \in \{i,i+1,\ldots,j-1\}$.

$f(i,j) = \min_k Q_k(i,j)$, $k = i,i+1,\ldots,j-1$.

$d(i,j)$ = k denotes the decision to test across the first k components on the next test, beginning in state $S(i,j)$.

$d^*(i,j)$ = the decision at $S(i,j)$ which results in $f(i,j)$, i.e., the optimal decision at $S(i,j)$.

2. Stage

We shall define the stage of the problem in terms of the system state. From the testing description in Section III, it follows that all possible system states can be enumerated in terms of the stage variable s as follows: $S(i,i+s)$ for $i = 1,2,\ldots,n-s$ and $s = 0,1,\ldots,n-1$. We shall let the stage correspond to the number s, so stages are numbered from zero to $n-1$, and at each stage s the possible system

states are $S(i,i+s)$ for $i = 1,2,\ldots,n-s$. For example, at stage zero the possible states are $S(1,1), S(2,2), \ldots, S(n,n)$. At stage three the possible states are $S(1,4), S(2,5), \ldots, S(n-3,n)$. Finally, at stage n-1 the only possible state is $S(1,n)$, which corresponds to the system state prior to any testing. The procedure to be employed is termed "backward recursive optimization" by Nemhauser [17]. Thus, while actual testing proceeds from stage n-1 to stage zero, solution of the problem begins with stage zero and works back to stage n-1.

3. Decision Variable

The decision variable at any state $S(i,j)$ is $d(i,j) \in \{i,i+1,\ldots,j-1\}$.

4. Objective Function

The objective is to minimize the expected number of tests required to locate the failed component, which can be stated as follows: Find $f(1,n) = \min_k [Q_k(1,n)] = \min E(N)$.

5. General Recursive Equation

Let the system be in state $S(t,m)$ at stage $s = m - t$. Then the general recursive equation is as follows:

$$f(t,m) = \min_k [Q_k(t,m)], \quad k \in \{t,t+1,\ldots,m-1\},$$

where $Q_k(t,m)$ is calculated as follows:

$$Q_k(t,m) = P[1 + f(t,k)] + (1-P)[1 + f(k+1,m)] , \qquad (1)$$

where $$P = \frac{\sum_{i=t}^{k} q_i}{\sum_{i=t}^{m} q_i} .$$

Note here that P is the probability that state $S(t,k)$ results from this test and $[1 + f(t,k)]$ is the minimum expected number of tests

required for completion of testing should S(t,k) result. The second term in (1) is similarly interpreted.

6. Initial Conditions

As we have stated above, testing is complete when the system state is S(i,i) for some i, since we then know component i to be failed. Thus f(i,i) = 0 for i = 1,2,...,n.

At state S(i,i+1) we have only to make one further test, thus f(i,i+1) = 1, and d*(i,i+1) = i for i = 1,...,n-1.

7. Solution Procedure

With the posterior probability of failure q_i given for each of the system's n components and the f values corresponding to all possible system states at stages zero and one given as shown above, begin with stage two. For each of the possible system states at stage two, which are of the form S(i,i+2) for i = 1,...,n-2, calculate $Q_k(i,i+2)$ for k = i,i+1. For each state S(i,i+2), $f(i,i+2) = \min_k[Q_k(i,i+2)]$, where $Q_k(i,i+2)$ is calculated from (1). With each state S(i,i+2) associate the optimal decision variable d*(i,i+2) which yields f(i,i+2).

Continue similarly for subsequent stages. Upon completion of stage n-1, a value for f(1,n) has been obtained. This is the minimum expected number of tests required to locate the failed component.

For the complete specification of the derived test plan, trace back through the stages and their associated d* values along sequential testing paths which terminate with state S(i,i) for all i.

8. Computer Program

A computer program, written in FORTRAN IV for use on the IBM 360-67 digital computer has been developed and used in the example cases included in Section V. A program listing and sample output is included following Appendix C.

B. SEQUENTIAL HALVING PROCEDURE

The sequential halving procedure is independent of the probabilities of failure attached to the system components. It can be described as follows: At any point in the procedure if m components remain to be tested, the next test should divide them as equally as possible. Thus, if m is even, the next test divides the components into two equal groups of $m/2$ each. If m is odd, the next test divides the components into two groups with $m/2 - 1/2$ in one group and $m/2 + 1/2$ in the other. For consistency, we will always take the smaller of the two groups to the left, e.g., if components 1-7 remain to be tested, we test next across components 1-3.

It is shown in Appendix C that this procedure is minimax. In the special case where the component probabilities of failure are all equal, this procedure satisfies the requirements of Sandelius' theorem given in Section II, and thus also yields the minimum expected number of tests required. In other cases, the expected number of tests required may not be minimized through this procedure.

C. THE INFORMATION THEORY FORMULATION

The information theory solution to the problem is a heuristic approach, based upon maximization of the expected reduction of uncertainty inherent in the system at each succeeding test. This procedure was mentioned in passing as a special case by Brulé, Johnson and Kletsky [22] in 1959.

The procedure may be simply stated as follows: Suppose the failed component is known to belong to a set of m components. The component posterior probabilities of failure given that one of the m components is failed, q_i, are known with $\sum_{i=1}^{m} q_i = 1$. Test next across the first

k components, choosing k such that $\left| \sum_{i=1}^{k} q_i - 1/2 \right|$ is a minimum.

A derivation of an expression for the computation of component posterior probabilities is contained in Appendix A. A derivation of the information theory procedure and an algorithm for its use are contained in Appendix B.

The attributes of the information theory solution in terms of the two criteria for optimality listed above is unclear, except to assert that the expected number of tests required will be at least that given by dynamic programming, and the maximum number of tests required will be at least that given by the sequential halving procedure. Further evaluation of this procedure is the purpose of the example cases considered in the next section.

V. EXAMPLE CASE RESULTS

Figures 5 through 19 summarize the salient features and results obtained in seventeen example cases.

A. KEY TO FIGURES 5 THROUGH 19

The following definitions apply to the entities of these figures:

N = the number of tests required to locate the failed component.

$E(N)$ = the expected value of N.

$V(N)$ = the variance of N.

N_{max} = the maximum number of tests required to locate the failed component.

p_i^o = the a priori reliability of component i.

q_i = the posterior probability component i is failed, given that the system has one failed component.

State = (i,j) denotes the set of components i through j inclusive is known to contain the failed component.

Dec = i denotes the decision to test across the first i components on this test.

Comp column denotes the component(s) which, if failed, would be located by the test.

pmf denotes the probability mass function values for N.

Complete test plan specifications are shown for both the dynamic programming and information theory solutions for all 20-component cases. Cases 15, 16 and 17 are 40-component cases, for which only summary results are included.

B. SEQUENTIAL HALVING TEST PLAN SPECIFICATION

The sequential halving test plan depends only on the number of components which comprise the system and is independent of the *a priori* component reliabilities assigned.

TEST	STATE	DEC	COMP
1	1,20	10	
2	1,10	5	
	11,20	15	
3	1, 5	2	
	6,10	7	
	11,15	12	
	16,20	17	
4	1, 2	1	1, 2
	3, 5	3	3
	6, 7	6	6, 7
	8,10	8	8
	11,12	11	11,12
	13,15	13	13
	16,17	16	16,17
	18,20	18	18
5	4, 5	4	4, 5
	9,10	9	9,10
	14,15	14	14,15
	19,20	19	19,20

20-Component Sequential Halving Test Plan

TABLE II

C. *A PRIORI* COMPONENT RELIABILITIES

A priori component reliabilities do not necessarily sum to unity, and are therefore not probability distributions.

Example case 1 is the 20-component equal probability case.

Component reliabilities are linearly arranged for cases 2 through 5, in order of increasing slope.

Component reliabilities are symmetric and follow the familiar "bell-shaped" curve in cases 6, 7 and 8 in order of increasing peakedness.

Cases 9, 10, 11, 15, 16 and 17 contain component reliabilities randomly selected between 0.7 and 1.0.

Component reliabilities are geometrically arranged for cases 12, 13 and 14 by $p^{o}_{20-(i-1)} = (p)^{i}$ for i=1,2,...,20. The p-values used are 0.95, 0.90 and 0.85 respectively.

D. COMPUTER PROGRAM VALIDATION

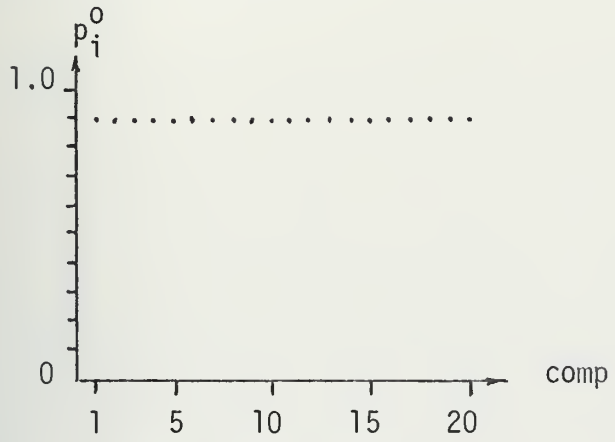
The computer program was validated first by running four and five-component cases for which the solution had been calculated manually.

For 10, 20, 40 and 80-component cases, it was validated by running the equal probability case for which the solution is given by Sandelius' theorem stated in Section II. For these cases the $E(N)$ value is accurate to at least four decimal places.

The solution given by the computer program does minimize $E(N)$, but does not indicate the existence of alternate optima. Such alternate solutions do in fact exist for the equal probability case. Thus, where the pharse "the dynamic programming solution" appears, perhaps the phrase "a dynamic programming solution" may be more appropriate.

EXAMPLE CASE 1

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.4000	0.2400	5
Info Theory	4.4000	0.2400	5
Seq Halving	4.4000	0.2400	5



Comp	p_i^o	q_i
1	.9000	.0500
2	.9000	.0500
3	.9000	.0500
4	.9000	.0500
5	.9000	.0500
6	.9000	.0500
7	.9000	.0500
8	.9000	.0500
9	.9000	.0500
10	.9000	.0500
11	.9000	.0500
12	.9000	.0500
13	.9000	.0500
14	.9000	.0500
15	.9000	.0500
16	.9000	.0500
17	.9000	.0500
18	.9000	.0500
19	.9000	.0500
20	.9000	.0500

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	10		0
2	1,10	5		0
	11,20	15		
3	1, 5	2		0
	6,10	7		
	11,15	12		
	16,20	17		
4	1, 2	1	1, 2	.6000
	3, 5	3	3	
	6, 7	6	6, 7	
	8,10	8	8	
	11,12	11	11,12	
	13,15	13	13	
	16,17	16	16,17	
	18,20	18	18	
5	4, 5	4	4, 5	.4000
	9,10	9	9,10	
	14,15	14	14,15	
	19,20	19	19,20	

INFORMATION THEORY

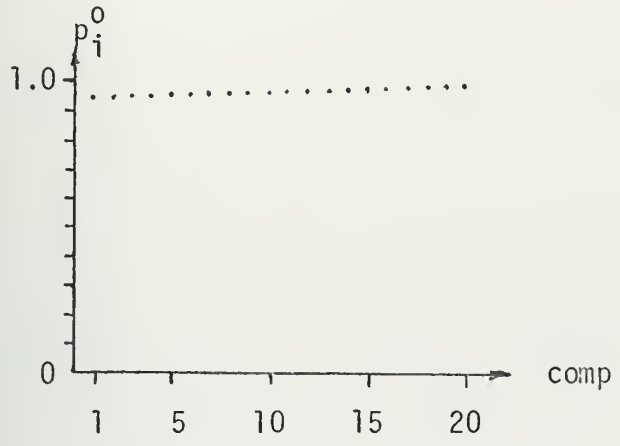
State	Dec	Comp	pmf

(Same sol'n as D. P.)

Figure 5

EXAMPLE CASE 2

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.3281	0.2204	5
Info Theory	4.3521	0.2281	5
Seq Halving	4.3811	0.2399	5



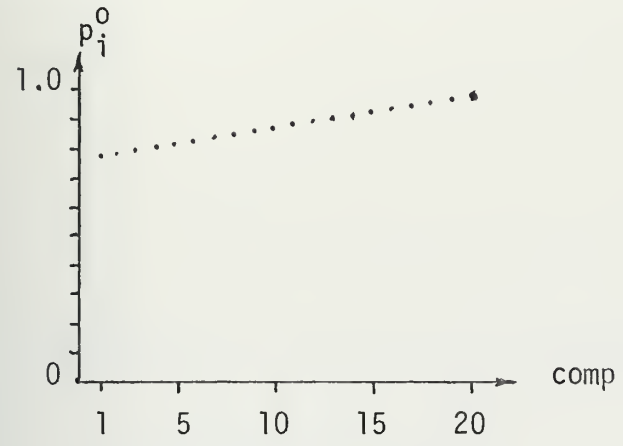
Comp	p_i^o	q_i
1	.9560	.0643
2	.9570	.0628
3	.9580	.0613
4	.9590	.0598
5	.9600	.0582
6	.9610	.0567
7	.9620	.0552
8	.9630	.0537
9	.9640	.0522
10	.9650	.0507
11	.9660	.0492
12	.9670	.0477
13	.9680	.0462
14	.9690	.0447
15	.9700	.0432
16	.9710	.0417
17	.9720	.0403
18	.9730	.0388
19	.9740	.0373
20	.9750	.0358

	DYNAMIC PROGRAMMING				INFORMATION THEORY			
Test	State	Dec	Comp	pmf	State	Dec	Comp	pmf
1	1,20	8		0	1,20	9		0
2	1, 8	4		0	1, 9	4		0
	9,20	12			10,20	14		
3	1, 4	2		0	1, 4	2		0
	5, 8	6			5, 9	6		
	9,12	10			10,14	11		
	13,20	16			15,20	17		
4	1, 2	1	1, 2	.6718	1, 2	1	1, 2	.6479
	3, 4	3	3, 4		3, 4	3	3, 4	
	5, 6	5	5, 6		5, 6	5	5, 6	
	7, 8	7	7, 8		7, 9	7	7	
	9,10	9	9,10		10,11	10	10,11	
	11,12	11	11,12		12,14	12	12	
	13,16	14			15,17	15	15	
	17,20	18			18,20	18	18	
5	13,14	13	13,14	.3280	8, 9	8	8, 9	.3519
	15,16	15	15,16		13,14	13	13,14	
	17,18	17	17,18		16,17	16	16,17	
	19,20	19	19,20		19,20	19	19,20	

Figure 6

EXAMPLE CASE 3

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.1446	0.6213	6
Info Theory	4.1691	0.7385	7
Seq Halving	4.3459	0.2263	5



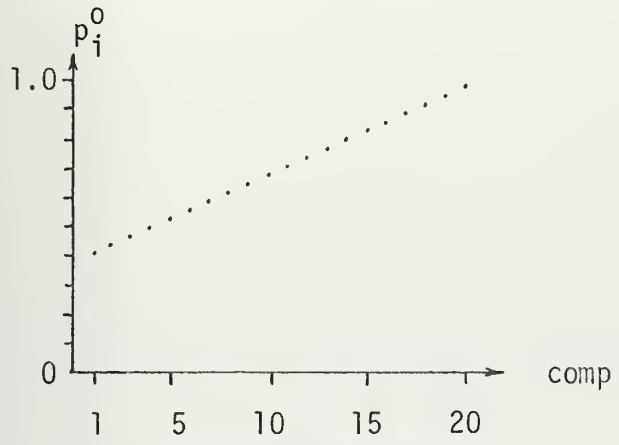
Comp	p_i^o	q_i
1	.7850	.0967
2	.7950	.0911
3	.8050	.0856
4	.8150	.0802
5	.8250	.0749
6	.8350	.0698
7	.8450	.0648
8	.8550	.0599
9	.8650	.0551
10	.8750	.0505
11	.8850	.0459
12	.8950	.0414
13	.9050	.0371
14	.9150	.0328
15	.9250	.0286
16	.9350	.0246
17	.9450	.0206
18	.9550	.0166
19	.9650	.0147
20	.9750	.0091

	DYNAMIC PROGRAMMING				INFORMATION THEORY			
Test	State	Dec	Comp	pmf	State	Dec	Comp	pmf
1	1,20	6		0	1,20	6		0
2	1, 6	2		0	1, 6	3		0
	7,20	10			7,20	10		
3	1, 2	1	1, 2	.1878	1, 3	2	3	.1658
	3, 6	4			4, 6	4	4	
	7,10	8			7,10	8		
	11,20	14			11,20	13		
4	3, 4	3	3, 4	.5408	1, 2	1	1, 2	.6087
	5, 6	5	5, 6		5, 6	5	5, 6	
	7, 8	7	7, 8		7, 8	7	7, 8	
	9,10	9	9,10		9,10	9	9,10	
	11,14	12			11,13	11	11	
	15,20	16			14,20	15		
5	11,12	11	11,12	.2104	12,13	12	12,13	.1399
	13,14	13	13,14		14,15	14	14,15	
	15,16	15	15,16		16,20	17		
	17,20	18						
6	17,18	17	17,18	.0610	16,17	16	16,17	.0618
	19,20	19	19,20		18,20	18	18	
7					19,20	19	19,20	.0238

Figure 7

EXAMPLE CASE 4

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	3.9152	0.8517	8
Info Theory	3.9152	0.8517	8
Seq Halving	4.3178	0.2168	5



Comp	p_i^o	q_i
1	.4050	.1330
2	.4350	.1176
3	.4650	.1042
4	.4950	.0924
5	.5250	.0819
6	.5550	.0726
7	.5850	.0642
8	.6150	.0567
9	.6450	.0498
10	.6750	.0436
11	.7050	.0379
12	.7350	.0326
13	.7650	.0278
14	.7950	.0233
15	.8250	.0192
16	.8550	.0154
17	.8850	.0118
18	.9150	.0084
19	.9450	.0053
20	.9750	.0023

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	Pmf
1	1,20	5		0
2	1, 5	2		0
	6,20	9		
3	1, 2	1	1, 2	.3548
	3, 5	3	3	
	6, 9	7		
	10,20	12		
4	4, 5	4	4, 5	.4612
	6, 7	6	6, 7	
	8, 9	8	8, 9	
	10,12	10	10	
	13,20	14		
5	11,12	11	11,12	.1216
	13,14	13	13,14	
	15,20	16		
6	15,16	15	15,16	.0464
	17,20	17	17	
7	18,20	18	18	.0084
8	19,20	19	19,20	.0076

INFORMATION THEORY

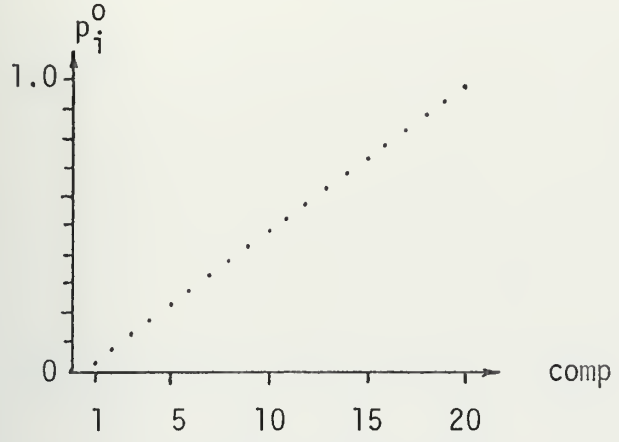
State	Dec	Comp	Pmf

(Same sol'n as D. P.)

Figure 8

EXAMPLE CASE 5

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	2.6504	3.7483	10
Info Theory	2.6504	3.7483	10
Seq Halving	4.1459	0.1266	5



Comp	p_i^o	q_i
1	.0250	.4925
2	.0750	.1557
3	.1250	.0884
4	.1750	.0595
5	.2250	.0435
6	.2750	.0333
7	.3250	.0262
8	.3750	.0210
9	.4250	.0171
10	.4750	.0140
11	.5250	.0114
12	.5750	.0093
13	.6250	.0076
14	.6750	.0061
15	.7250	.0048
16	.7750	.0037
17	.8250	.0027
18	.8750	.0018
19	.9250	.0010
20	.9750	.0003

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	1	1	.4925
2	2,20	3		
3	2, 3	2	2, 3	.2441
	4,20	6		
4	4, 6	4	4	.0595
	7,20	9		
5	5, 6	5	5, 6	.1030
	7, 9	7	7	
	10,20	12		
6	8, 9	8	8, 9	.0521
	10,12	10	10	
	13,20	14		
7	11,12	11	11,12	.0344
	13,14	13	13,14	
	15,20	16		
8	15,16	15	15,16	.0112
	17,20	17	17	
9	18,20	18	18	.0018
10	19,20	19	19,20	.0013

INFORMATION THEORY

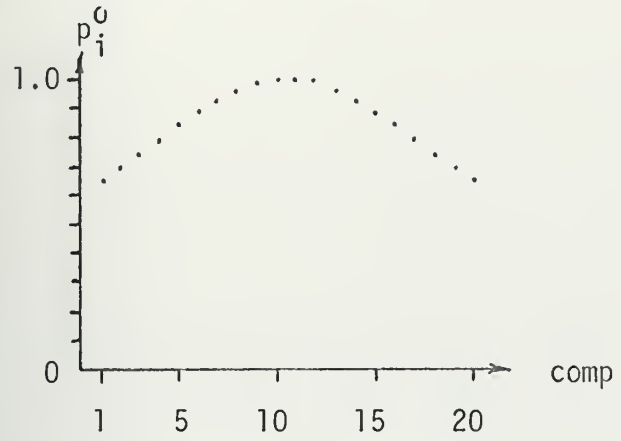
State	Dec	Comp	pmf

(Same sol'n as D.P.)

Figure 9

EXAMPLE CASE 6

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	3.8296	1.0407	10
Info Theory	3.8346	1.0340	8
Seq Halving	4.4023	0.2405	5



Comp	p_i^o	q_i
1	.6656	.1165
2	.6940	.1022
3	.7295	.0860
4	.7714	.0687
5	.8179	.0516
6	.8661	.0359
7	.9123	.0223
8	.9521	.0117
9	.9814	.0044
10	.9970	.0007
11	.9970	.0007
12	.9814	.0044
13	.9521	.0117
14	.9123	.0223
15	.8661	.0359
16	.8179	.0516
17	.7714	.0687
18	.7295	.0860
19	.6940	.1022
20	.6656	.1165

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	12		0
2	1,12	2		0
	13,20	18		
3	1, 2	1	1, 2	.4374
	3,12	4		
	13,18	16		
	19,20	19	19,20	
4	3, 4	3	3, 4	.4126
	5,12	5	5	
	13,16	15	16	
	17,18	17	17,18	
5	6,12	6	6	.0718
	13,15	14	15	
6	7,12	7	7	.0563
	13,14	13	13,14	
7	8,12	8	8	.0117
8	9,12	11	12	.0044
9	9,11	9	9	.0044
10	10,11	10	10,11	.0014

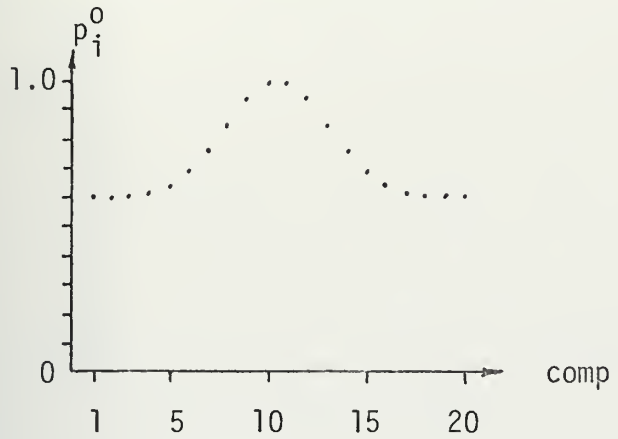
INFORMATION THEORY

State	Dec	Comp	pmf
1,20	10		0
1,10	2		0
11,20	18		
1, 2	1	1, 2	.4374
3,10	4		
11,18	16		
19,20	19	19,20	
3, 4	3	3, 4	.4126
5,10	5	5	
11,16	15	16	
17,18	17	17,18	
6,10	6	6	.0718
11,15	14	15	
7,10	7	7	.0446
11,14	13	14	
8,10	8	8	.0234
11,13	12	13	
9,10	9	9,10	.0102
11,12	11	11,12	

Figure 10

EXAMPLE CASE 7

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.0556	0.3015	8
Info Theory	4.0680	0.5738	7
Seq Halving	4.4021	0.2364	5



Comp	p_i^o	q_i
1	.6003	.0781
2	.6012	.0778
3	.6044	.0768
4	.6136	.0739
5	.6355	.0673
6	.6790	.0555
7	.7497	.0392
8	.8420	.0220
9	.9332	.0084
10	.9910	.0011
11	.9910	.0011
12	.9332	.0084
13	.8420	.0220
14	.7497	.0392
15	.6790	.0555
16	.6355	.0673
17	.6136	.0739
18	.6044	.0768
19	.6012	.0778
20	.6003	.0781

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	7		0
2	1, 7	3		0
	8,20	16		
3	1, 3	1	1	.0781
	4, 7	5		
	8,16	14		
	17,20	18		
4	2, 3	2	2, 3	.8199
	4, 5	4	4, 5	
	6, 7	6	6, 7	
	8,14	12		
	15,16	15	15,16	
	17,18	17	17,18	
	19,20	19	19,20	
5	8,12	8	8	.0832
	13,14	13	13,14	
6	9,12	11	12	.0084
7	9,11	9	9	.0084
8	10,11	10	10,11	.0022

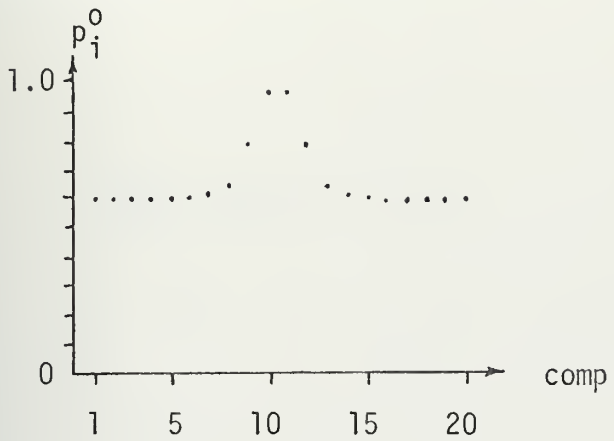
INFORMATION THEORY

State	Dec	Comp	pmf
1,20	10		0
1,10	3		0
11,20	17		
1, 3	1	1	.1562
4,10	5		
11,17	15		
18,20	19	20	
2, 3	2	2, 3	.7026
4, 5	4	4, 5	
6,10	6	6	
11,15	14	15	
16,17	16	16,17	
18,19	18	18,19	
7,10	7	7	.0784
11,14	13	14	
8,10	8	8	.0440
11,13	12	13	
9,10	9	9,10	.0190
11,12	11	11,12	

Figure 11

EXAMPLE CASE 8

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.1859	0.2347	7
Info Theory	4.2036	0.2689	6
Seq Halving	4.3906	0.2380	5



Comp	p_i^o	q_i
1	.6000	.0610
2	.6000	.0610
3	.6000	.0610
4	.6000	.0610
5	.6000	.0610
6	.6006	.0609
7	.6079	.0590
8	.6540	.0484
9	.7942	.0237
10	.9683	.0030
11	.9683	.0030
12	.7952	.0237
13	.6540	.0484
14	.6079	.0590
15	.6006	.0609
16	.6000	.0610
17	.6000	.0610
18	.6000	.0610
19	.6000	.0610
20	.6000	.0610

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	11		0
2	1,11	4		0
	12,20	16		
3	1, 4	2		0
	5,11	6		
	12,16	14		
	17,20	18		
4	1, 2	1	1, 2	.8498
	3, 4	3	3, 4	
	5, 6	5	5, 6	
	7,11	7	7	
	12,14	13	14	
	15,16	15	15,16	
	17,18	17	17,18	
	19,20	19	19,20	
5	8,11	8	8	.1205
	12,13	12	12,13	
6	9,11	9	9	.0237
7	10,11	10	10,11	.0060

INFORMATION THEORY

State	Dec	Comp	pmf
1,20	10		0
1,10	4		0
11,20	16		
1, 4	2		0
5,10	6		
11,16	14		
17,20	18		
1, 2	1	1, 2	.8498
3, 4	3	3, 4	
5, 6	5	5, 6	
7,10	7	7	
11,14	13	14	
15,16	15	15,16	
17,18	17	17,18	
19,20	19	19,20	
8,10	8	8	.0968
11,13	12	13	
9,10	9	9,10	.0534
11,12	11	11,12	

Figure 12

EXAMPLE CASE 9

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	3.9534	1.1439	6
Info Theory	4.0522	0.9017	6
Seq Halving	4.2000	0.1660	5

(p_i^o random 0.7 to 1.0)

Comp	p_i^o	q_i
1	.8153	.0738
2	.9192	.0286
3	.9157	.0300
4	.9215	.0278
5	.9842	.0052
6	.7006	.1392
7	.7607	.1025
8	.9072	.0333
9	.9501	.0171
10	.9309	.0242
11	.7290	.1211
12	.9197	.0284
13	.9396	.0209
14	.9733	.0089
15	.9762	.0079
16	.9124	.0313
17	.7229	.1249
18	.8347	.0645
19	.8963	.0377
20	.8183	.0724

	DYNAMIC PROGRAMMING				INFORMATION THEORY			
Test	State	Dec	Comp	pmf	State	Dec	Comp	pmf
1	1,20	10		0	1,20	10		0
2	1,10	6		0	1,10	6		0
	11,20	16			11,20	16		
3	1, 6	5	6	.4877	1, 6	4		.3485
	7,10	7	7		7,10	7	7	
	11,16	11	11		11,16	11	11	
	17,20	17	17		17,20	17	17	
4	1, 5	1	1	.1795	1, 4	1	1	.3239
	8,10	8	8		5, 6	5	5, 6	
	12,16	13			8,10	8	8	
	18,20	19	20		12,16	13		
					18,20	19	20	
5	2, 5	3		.2241	2, 4	2	2	.2527
	9,10	9	9,10		9,10	9	9,10	
	12,13	12	12,13		12,13	12	12,13	
	14,16	15	16		14,16	15	16	
	18,19	18	18,19		18,19	18	18,19	
6	2, 3	2	2, 3	.1084	3, 4	3	3, 4	.0746
	4, 5	4	4, 5		14,15	14	14,15	
	14,15	14	14,15					

Figure 13

EXAMPLE CASE 10

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.1443	0.6239	5
Info Theory	4.2302	0.4432	5
Seq Halving	4.3768	0.2348	5

(p_i^o random 0.7 to 1.0)

Comp	p_i^o	q_i
1	.8823	.0435
2	.8752	.0465
3	.7104	.1330
4	.8582	.0539
5	.8763	.0461
6	.8791	.0449
7	.9587	.0141
8	.7357	.1172
9	.9094	.0325
10	.9482	.0178
11	.8872	.0415
12	.8397	.0623
13	.9352	.0226
14	.8539	.0558
15	.9726	.0092
16	.9273	.0256
17	.8568	.0545
18	.9490	.0175
19	.7410	.1140
20	.8730	.0475

	DYNAMIC PROGRAMMING				INFORMATION THEORY			
Test	State	Dec	Comp	pmf	State	Dec	Comp	pmf
1	1,20	8		0	1,20	8		0
2	1, 8	3		0	1, 8	3		0
	9,20	14			9,20	15		
3	1, 3	2	3	.2502	1, 3	2	3	.1330
	4, 8	7	8		4, 8	6		
	9,14	11			9,15	11		
	15,20	18			16,20	18		
4	1, 2	1	1, 2	.3553	1, 2	1	1, 2	.5038
	4, 7	5			4, 6	4	4	
	9,11	10	11		7, 8	7	7, 8	
	12,14	12	12		9,11	10	11	
	15,18	16			12,15	13		
	19,20	19	19,20		16,18	16	16	
					19,20	19	19,20	
5	4, 5	4	4, 5	.3945	5, 6	5	5, 6	.3632
	6, 7	6	6, 7		9,10	9	9,10	
	9,10	9	9,10		12,13	12	12,13	
	13,14	13	13,14		14,15	14	14,15	
	15,16	15	15,16		17,18	17	17,18	
	17,18	17	17,18					

Figure 14

EXAMPLE CASE 11

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.2404	0.6364	6
Info Theory	4.2875	0.3620	6
Seq Halving	4.4464	0.2451	5

(p_i^o random 0.7 to 1.0)

Comp	p_i^o	q_i
1	.9642	.0092
2	.9430	.0149
3	.7734	.0723
4	.8860	.0318
5	.7317	.0689
6	.8126	.0569
7	.8276	.0514
8	.7076	.1020
9	.8164	.0555
10	.7639	.0763
11	.9887	.0028
12	.7099	.1008
13	.8897	.0306
14	.7723	.0728
15	.7745	.0718
16	.8936	.0294
17	.8467	.0447
18	.8632	.0391
19	.8461	.0449
20	.9113	.0240

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	11		0
2	1,11	7		0
	12,20	14		
3	1, 7	5		.2028
	8,11	8	8	
	12,14	12	12	
	15,20	17		
4	1, 5	3		.3781
	6, 7	6	6, 7	
	9,11	9	9	
	13,14	13	13,14	
	15,17	15	15	
	18,20	18	18	
5	1, 3	2	3	.3951
	4, 5	4	4, 5	
	10,11	10	10,11	
	16,17	16	16,17	
	19,20	19	19,20	
6	1, 2	1	1, 2	.0241

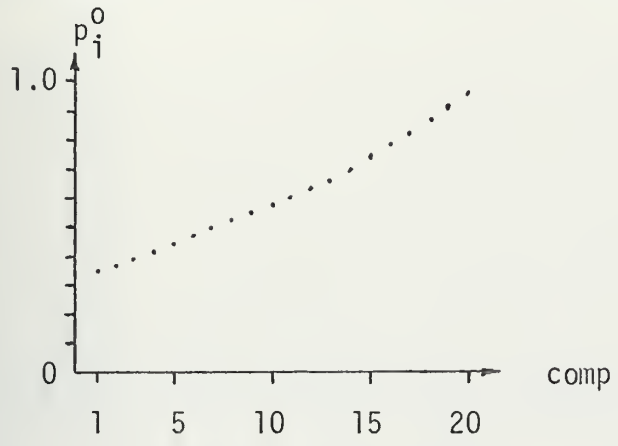
INFORMATION THEORY

State	Dec	Comp	pmf
1,20	9		0
1, 9	6		0
10,20	14		
1, 6	4		.0555
7, 9	8	9	
10,14	12		
15,20	17		
1, 4	3	4	.6261
5, 6	5	5, 6	
7, 8	7	7, 8	
10,12	11	12	
13,14	13	13,14	
15,17	15	15	
18,20	18	18	
1, 3	2	3	.2944
10,11	10	10,11	
16,17	16	16,17	
19,20	19	19,20	
1, 2	1	1, 2	.0241

Figure 15

EXAMPLE CASE 12

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	4.0089	0.9336	8
Info Theory	4.0089	0.9336	8
Seq Halving	4.2981	0.5028	5



Comp	p_i^o	q_i
1	.3584	.1134
2	.3773	.1045
3	.3972	.0961
4	.4181	.0881
5	.4401	.0806
6	.4633	.0734
7	.4877	.0665
8	.5134	.0600
9	.5404	.0539
10	.5688	.0480
11	.5987	.0424
12	.6302	.0372
13	.6634	.0321
14	.6983	.0274
15	.7351	.0228
16	.7738	.0185
17	.8145	.0144
18	.8574	.0105
19	.9025	.0068
20	.9500	.0033

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	5		0
2	1, 5	2		0
	6,20	9		
3	1, 2	1	1, 2	.3140
	3, 5	3	3	
	6, 9	7		
	10,20	12		
4	4, 5	4	4, 5	.4705
	6, 7	6	6, 7	
	8, 9	8	8, 9	
	10,12	10	10	
	13,20	14		
5	11,12	11	11,12	.1391
	13,14	13	13,14	
	15,20	16		
6	15,16	15	15,16	.0557
	17,20	17	17	
7	18,20	18	18	.0105
8	19,20	19	19,20	.0101

INFORMATION THEORY

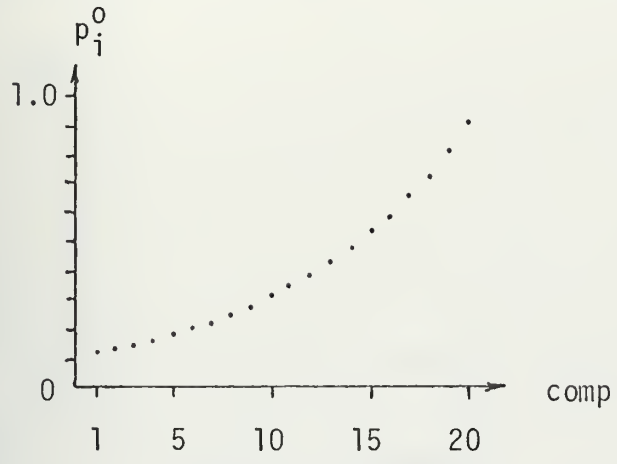
State	Dec	Comp	pmf

(Same sol'n as D. P.)

Figure 16

EXAMPLE CASE 13

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	3.8613	1.1091	8
Info Theory	3.8613	1.1091	8
Seq Halving	4.3146	0.2156	5



Comp	p_i^o	q_i
1	.1215	.1383
2	.1350	.1226
3	.1500	.1084
4	.1667	.0956
5	.1852	.0842
6	.2058	.0738
7	.2287	.0645
8	.2541	.0562
9	.2823	.0486
10	.3137	.0418
11	.3486	.0357
12	.3875	.0302
13	.4306	.0253
14	.4784	.0209
15	.5315	.0169
16	.5905	.0133
17	.6561	.0100
18	.7290	.0071
19	.8100	.0045
20	.9000	.0021

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	4		0
2	1, 4	2		0
	5,20	8		
3	1, 2	1	1, 2	.4649
	3, 4	3	3, 4	
	5, 8	6		
	9,20	11		
4	5, 6	5	5, 6	.3273
	7, 8	7	7, 8	
	9,11	9	9	
	12,20	13		
5	10,11	10	10,11	.1330
	12,13	12	12,13	
	14,20	15		
6	14,15	14	14,15	.0378
	16,20	17		
7	16,17	16	16,17	.0304
	18,20	18	18	
8	19,20	19	19,20	.0066

INFORMATION THEORY

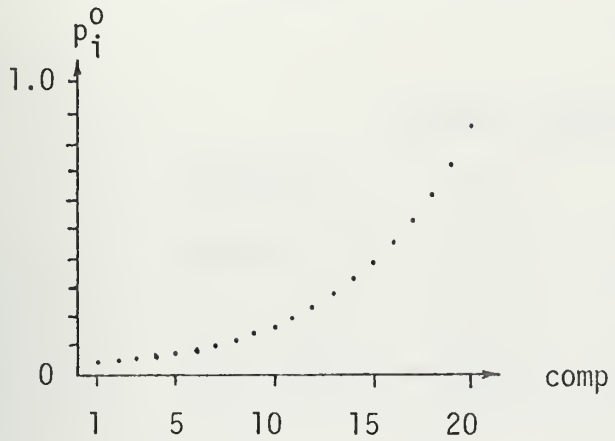
State	Dec	Comp	pmf

(Same sol'n as D. P.)

Figure 17

EXAMPLE CASE 14

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	3.6845	1.1853	9
Info Theory	3.6845	1.1853	9
Seq Halving	4.2934	0.2073	5



Comp	p_i^o	q_i
1	.0388	.1705
2	.0456	.1441
3	.0537	.1213
4	.0632	.1020
5	.0743	.0858
6	.0874	.0719
7	.1028	.0601
8	.1209	.0501
9	.1422	.0415
10	.1673	.0343
11	.1968	.0281
12	.2315	.0229
13	.2724	.0184
14	.3205	.0146
15	.3771	.0114
16	.4437	.0086
17	.5220	.0063
18	.6141	.0043
19	.7225	.0026
20	.8500	.0012

DYNAMIC PROGRAMMING

Test	State	Dec	Comp	pmf
1	1,20	4		0
2	1, 4	2		0
	5,20	7		
3	1, 2	1	1, 2	.6237
	3, 4	3	3, 4	
	5, 7	5	5	
	8,20	10		
4	6, 7	6	6, 7	.1821
	8,10	8	8	
	11,20	12		
5	9,10	9	9,10	.1268
	11,12	11	11,12	
	13,20	14		
6	13,14	13	13,14	.0330
	15,20	16		
7	15,16	15	15,16	.0263
	17,20	17	17	
8	18,20	18	18	.0043
9	19,20	19	19,20	.0038

INFORMATION THEORY

State	Dec	Comp	pmf

(Same sol'n as D. P.)

Figure 18

EXAMPLE CASE 15

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	5.0120	0.9941	7
Info Theory	5.0761	0.5449	7
Seq Halving	5.2636	0.1881	6

EXAMPLE CASE 16

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	5.1987	0.6088	7
Info Theory	5.2654	0.2816	7
Seq Halving	5.4600	0.2484	6

EXAMPLE CASE 17

Procedure	E(N)	V(N)	N_{max}
Dynamic Prog	5.1213	0.7107	7
Info Theory	5.2038	0.4255	7
Seq Halving	5.4741	0.2433	6

Summary Results for Example Cases 15, 16 and 17

Figure 19

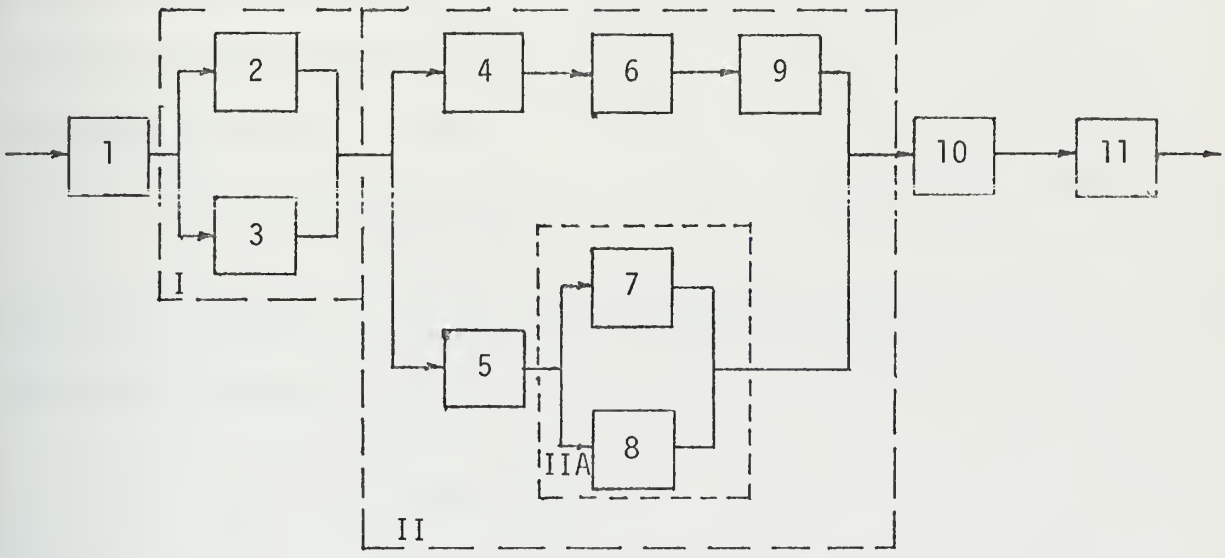
VI. CONCLUSIONS

A. MODEL APPLICABILITY

The assumptions that the system components are in series, and that exactly one failure occurs are not as restrictive as they appear at first glance.

1. Series-Parallel Systems

In many instances a complex system consisting of series and parallel networks can be reduced to an equivalent series system. Consider the system shown in Figure 20. Let R_i be the a priori



Series-Parallel System

Figure 20

reliability of component i for $i=1,2,\ldots,11$. Suppose the components work or fail independently of each other.

The reliability of an n-component series system is

$$R_s = \prod_{i=1}^{n} R_i \ ,$$

since the probability the system works is equivalent to the probability that all components work. This is commonly called the "product rule".

Since $(1-R_i)$ is the probability component i is failed, $\prod_{i=1}^{n}(1-R_i)$ is the probability that all components are failed. All components of an n-component parallel system must fail to cause system failure, thus

$$R_p = 1 - \prod_{i=1}^{n}(1-R_i)$$

is the reliability of a parallel system.

With the above expressions for R_s and R_p, the system in Figure 20 may now be reduced to an equivalent series system. First, components 2 and 3 form a parallel system. They can be replaced by an equivalent component I with

$$R_I = 1 - (1-R_2)(1-R_3).$$

Similarly, components 7 and 8 can be replaced by component IIA in series with component 5 with

$$R_{IIA} = 1 - (1-R_7)(1-R_8).$$

Now, components 4, 6 and 9, and components 5 and IIA form two series systems. Thus the reliabilities of the upper and lower branches of II are $R_4R_6R_9$ and R_5R_{IIA} respectively. These two branches form a parallel system, so they can be replaced by the equivalent component II with

$$R_{II} = 1 - (1-R_4R_6R_9)(1-R_5R_{IIA}).$$

The system has now been reduced to an equivalent series system consisting of components 1, I, II, 10 and 11. The model may now be employed to derive a test plan for the equivalent system.

If the failure is located to lie in component I, both components 2 and 3 are then known to be failed. Should component II be failed, its upper and lower branches are then treated as failed series systems for which the model is applicable.

The above procedure to reduce the system to an equivalent series system requires a "grouping" of individual components which in turn constrains some properties of the total system test plan. Thus, the three solution procedures are not claimed to be optimal.

2. Component Dependence

The fact that some inter-dependence exists among the components of most systems is not to be denied. However, one would expect such relationships to be difficult to define at best. It is therefore questionable whether the cost required for a correlation study of multiple failures would be justified in terms of improvement of results.

Two or more components may be dependent to such a degree that the independence assumption is no longer an acceptable approximation. In this case one may still be able to convert the system to consist only of independent components. Each set of dependent components may be combined into one equivalent component. These equivalent components then operate independently in the system. This aggregation over dependent components creates a system with independent components, but also creates the above-mentioned "grouping" constraints on the test plan. This tradeoff is a factor in the determination of the appropriate level of aggregation.

Once the components are assumed to function independently, even moderate system reliabilities yield small probabilities for multiple component failures. Consider the case in which the component reliabilities are all equal, i.e., $R_i = R$ for $i=1,\ldots,n$. Denote the number of failures by X, and note that X is distributed Binomial(n,1-R). Then let

$$P_1 = P\{\text{one failure}|\text{system down}\} = \frac{P(X=1)}{1-P(X=0)}$$

and, $P\{\text{multiple failures}|\text{system down}\} = 1 - P_1 = Q_1$

System reliability, $R_s = R^n = P(X=0)$. These quantities for various combinations of n and R are shown in Table III. This shows that,

Example	n	R	P_1	Q_1	R_s
1	5	0.950	0.90	0.10	0.77
2	5	0.975	0.95	0.05	0.88
3	10	0.950	0.79	0.21	0.60
4	10	0.975	0.89	0.11	0.78

Conditional Failure Probabilities
for Equal Reliability Case

Table III

for example one, the ratio of P_1 to $1-P_1$ is nine to one. An increase in component reliabilities (examples one and two) increases this ratio. An increase in the number of components, maintaining the same component reliability (examples one and three) reduces the ratio, but system reliability is also degraded. With an attendant increase in component reliability (examples one and four) to retain approximately

the same system reliability, the large ratio is preserved. System reliabilities of operational equipment are generally high. Thus the assumption of a single failed component seems reasonable.

3. Constrained Maximum Number of Tests

Should there exist a constraint on the maximum number of tests allowed, e.g., for budgetary or physical reasons, this model is not appropriate. The author understands that a dynamic programming formulation for this constrained problem has been developed, but that computer CPU time and core storage requirements limit its application to small systems.

For larger systems then, this model can be used to generate three solutions from which to choose. Given the maximum number of tests n, select that solution from the three for which $P(N \leq n) = \sum_{i=1}^{n} P(N=i)$ is the largest.

B. MERITS OF THE INFORMATION THEORY PROCEDURE

1. Expected Number of Tests Required

For the criterion of minimizing E(N), the information theory solution is inferior to that generated by dynamic programming. The crucial question is whether the ease of computation of the information theory solution can offset its non-optimality.

Let $E(N)_1 = E(N)$ given by dynamic programming,

$E(N)_2 = E(N)$ given by information theory, and

$E(N)_3 = E(N)$ given by sequential halving.

Define $PD_i = \frac{E(N)_i - E(N)_1}{E(N)_1}(100)$ = the percent degradation from the optimal solution, for $i = 2,3$.

Over the results of all 20-component example cases, the average $PD_2 = 0.6\%$, with the maximum $PD_2 = 2.5\%$ on case 9. Over these same cases, the average $PD_3 = 9.6\%$, with the maximum $PD_3 = 56.4\%$ on case 2.

The 20-component cases for which the component reliabilities were selected randomly between 0.7 and 1.0 yielded the largest PD_2 values. Three 40-component cases with component reliabilities similarly selected were also run. These results, summarized in Table IV, indicate that in terms of the E(N) criterion, the information theory solution is quite good, at least for up to forty components. Further, the PD_2 values remained the same order of magnitude when the number of components was doubled from twenty to forty.

Case	Nr Comps	PD_2	PD_3
9	20	2.5	6.3
10	20	2.1	5.6
11	20	1.1	4.9
15	40	1.3	5.0
16	40	1.3	5.0
17	40	1.6	6.9

Random (0.7-1.0) Case Results

Table IV

2. Maximum Number of Tests Required

Of all example cases conducted, only in case 5 did N_{max} for the information theory solution exceed that of the dynamic programming solution. In cases 6, 7 and 8 (all with component reliabilities symmetrically arranged), N_{max} for the information theory solution was strictly less than that of dynamic programming. For these cases, the

first decision $d^*(1,20) = 10$ appeared to be primarily responsible for this result. Note that for symmetrically arranged component reliabilities, information theory will always split them in half on the first test. One may conjecture here that N_{max} for the information theory solution is likely to be less than or equal to that of dynamic programming.

3. Computation Costs

The computer program written for the dynamic programming solution to the example cases is not claimed to be optimum in terms of the CPU time or the core storage required, yet it is believed to be representative of the CPU time and core storage requirements for this problem solution technique.

To obtain an estimate of the relationship between CPU time required and the number of components in the system, the library subroutine SETIME/GETIME was employed. Three cases each were run for 10, 20 and 40 components and one 80-component case, all with component reliabilities randomly selected from 0.7 to 1.0. The CPU times required for both calculations and output of results were recorded as shown in Table V. The time required for solution calculations heavily

Nr Comps	Calculation Time(sec)	Avg	Output Time (sec)	Avg
10	0.0346		0.0300	
10	0.0334	0.0340	0.0316	0.0310
10	0.0341		0.0313	
20	0.2514		0.0592	
20	0.2410	0.2457	0.0543	0.0570
20	0.2448		0.0574	
40	1.9635		0.1234	
40	1.9657	1.9504	0.1191	0.1205
40	1.9320		0.1191	
80	14.5887	14.5887	0.2161	0.2161

CPU Time Requirements

Table V

dominates the total time required as the number of components is increased. Further, as the number of components is doubled, the CPU time required for these calculations increases by a multiplicative factor of between seven and eight. Accordingly, projected CPU time requirements for larger cases are shown in Table VI.

Nr Comps	Projected Calculation Time
160	102-117 sec.
320	12-17 min.
640	84-136 min.

Projected CPU Time Requirements

Table VI

Core storage requirements and projected storage requirements for larger cases are shown in Table VII.

Nr Comps	Core Storage (Bytes x 1000)	Nr Comps	Projected Core Storage (Bytes x 1000)
10	45.0	160	251
20	47.5	320	868
40	57.5	640	3,331
80	96.5		

Core Storage Requirements

Table VII

As is apparent from Tables VI and VII, core storage requirements would be the controlling factor in limiting the maximum number of components which could be considered in the dynamic programming computer program.

The information theory solution procedure is simple enough to require only basic desk calculator skill. Forty-component cases required about an hour and a half for the author on a desk calculator (not including the time required to calculate q-values). The practical limits of this solution method are estimated to be at about 100-160 components. For systems substantially larger the author believes that a computer program could be developed which would require far less core storage than the dynamic programming solution. Thus, for large systems, it may well be the case that information theory and sequential halving may provide the "best" alternative solutions.

APPENDIX A

SYSTEM POSTERIOR PROBABILITIES

A. STATEMENT OF THE PROBLEM

Given the a priori probabilities that the components are working, find the corresponding posterior probabilities given the system has exactly one failed component.

B. DERIVATION

Define the following:

p_i^o = a priori probability component i is working.

$q_i^o = 1 - p_i^o$ = a priori probability component i is failed.

$p_i = 1 - q_i$ = posterior probability component i is working given the system has exactly one failed component.

$\{A_i\}$ = the event component i is working. $\{\overline{A}_i\}$ is the complementary event.

$\{B\}$ = the event exactly one component is failed.

We wish to find $p_i = P(A_i|B)$, $i = 1,2,\ldots,n$.

$$p_i = P(A_i|B) = 1 - P(\overline{A}_i|B) = 1 - \frac{P(\overline{A}_i \cap B)}{P(B)}$$

$$= 1 - \frac{q_i^o \prod_{j\neq i} p_j^o}{\sum_{k=1}^{n} q_k^o \prod_{j\neq k} p_j^o}$$

$$\therefore \quad q_i = \frac{q_i^o \prod_{j\neq i} p_j^o}{\sum_{k=1}^{n} q_k^o \prod_{j\neq k} p_j^o} \quad , \tag{1}$$

and we have that $\sum_{i=1}^{n} q_i = 1$.

C. COMPUTATIONAL FORM

To apply the information theory procedure we must first find q_i for $i = 1,2,\ldots,n$. We seek a form of (1) which is more suitable for hand calculations:

$$q_i = \frac{q_i^o \prod_{j \neq i} p_j^o}{\sum_{k=1}^{n} q_k^o \prod_{j \neq k} p_j^o} = \frac{q_i^o \prod_{j \neq i} p_j^o}{\sum_{k=1}^{n} q_k^o / p_k^o (\prod_j p_j^o)} = \frac{q_i^o / p_i^o (\prod_j p_j^o)}{(\prod_j p_j^o) \sum_{k=1}^{n} q_k^o / p_k^o}$$

$$q_i = \frac{q_i^o / p_i^o}{\sum_{k=1}^{n} q_k^o / p_k^o} \qquad (2)$$

D. ALGORITHM FOR COMPUTATION

1. Calculate q_i^o / p_i^o for each component $(q_i^o = 1 - p_i^o)$
2. Sum results of step (1).
3. Divide each q_i^o / p_i^o from step (1) by the result of step (2).

APPENDIX B

INFORMATION THEORY PROCEDURE

A. DERIVATION

Define the following:

U_o = the uncertainty (entropy) present in an n-component system known to have exactly one failed component.

U_m = the uncertainty remaining after one test across the first m components.

q_i = posterior probability component i is failed given the system has exactly one failed component. $\sum_{i=1}^{n} q_i = 1$.

i^* = the number of the failed component.

Khinchin[14] offers the following rationale for the uncertainty (entropy) concept in probability theory. A complete system of events is a mutually exclusive and exhaustive set of events $A_1, A_2, \ldots, A_n$. Given the events of a complete system, together with their probabilities of occurrence $q_1, q_2, \ldots, q_n$ $(q_i \geq 0, \sum_{i=1}^{n} q_i = 1)$, we have a finite scheme. For example, one toss of a true die yields the finite scheme $\{A_1, \ldots, A_6\}$ with $A_i = \{\text{die shows } i\}$, hence $q_i = 1/6$ for all i. Any finite scheme describes a state of uncertainty in that we don't know which of the events $A_1, \ldots, A_n$ will be realized.

Define the quantity $U(q_1, \ldots, q_n) = -\sum_{i=1}^{n} q_i \ln q_i$ as a measure of the uncertainty inherent in a finite scheme, $A_1, \ldots, A_n$, taking $q_i \ln q_i = 0$ if $q_i = 0$. In our problem, $A_i = \{\text{component } i \text{ is failed}\}$, hence $U_o = -\sum_{i=1}^{n} q_i \ln q_i$.

We wish to maximize the expected reduction in uncertainty at each test. Thus, the problem is:

$$\max \quad E(U_o - U_m)$$

$$\text{s.t. } m = 1,2,\ldots,n-1$$

$E(U_o - U_m) = E(U_o) - E(U_m) = U_o - E(U_m)$, since U_o is a constant.

$$E(U_m) = P(1 \le i^* \le m)E(U_m \mid 1 \le i^* \le m) + P(m+1 \le i^* \le n)E(U_m \mid m+1 \le i^* \le n)$$

$$= \left(\sum_{i=1}^{m} q_i\right) (-1) \sum_{j=1}^{m} \left(\frac{q_j}{\sum_{i=1}^{m} q_i}\right) \ln \left(\frac{q_j}{\sum_{i=1}^{m} q_i}\right) +$$

$$\left(\sum_{i=m+1}^{n} q_i\right) (-1) \sum_{j=m+1}^{n} \left(\frac{q_j}{\sum_{i=m+1}^{n} q_i}\right) \ln \left(\frac{q_j}{\sum_{i=m+1}^{n} q_i}\right) .$$

So $$E(U_o - U_m) = -\sum_{i=1}^{n} q_i \ln q_i - \left\{ \left(\sum_{i=1}^{m} q_i\right) (-1) \sum_{j=1}^{m} \left(\frac{q_j}{\sum_{i=1}^{m} q_i}\right) \ln \left(\frac{q_j}{\sum_{i=1}^{m} q_i}\right) + \right.$$

$$\left. \left(\sum_{i=m+1}^{n} q_i\right) (-1) \sum_{j=m+1}^{n} \left(\frac{q_j}{\sum_{i=m+1}^{n} q_i}\right) \ln \left(\frac{q_j}{\sum_{i=m+1}^{n} q_i}\right) \right\}$$

$$E(U_o - U_m) = -\sum_{i=1}^{n} q_i \ln q_i + \sum_{j=1}^{m} q_j \ln \left(\frac{q_j}{\sum_{i=1}^{m} q_i}\right) + \sum_{j=m+1}^{n} q_j \ln \left(\frac{q_j}{\sum_{i=m+1}^{n} q_i}\right)$$

Since $\sum_{i=1}^{n} q_i \ln q_i$ is a constant, we can simply

$$\max Q_m = \sum_{j=1}^{m} q_j \ln \left(\frac{q_j}{\sum_{i=1}^{m} q_i}\right) + \sum_{j=m+1}^{n} q_j \ln \left(\frac{q_j}{\sum_{i=m+1}^{n} q_i}\right)$$

$$\text{s. t. } m = 1,2,\ldots,n-1$$

But $Q_m = \sum_{j=1}^{m} q_j \left[\ln q_j - \ln \left(\sum_{i=1}^{m} q_i \right) \right] + \sum_{j=m+1}^{n} q_j \left[\ln q_j - \ln \left(\sum_{i=m+1}^{n} q_i \right) \right]$

$$= \sum_{j=1}^{n} q_j \ln q_j - \sum_{j=1}^{m} q_j \ln \left(\sum_{i=1}^{m} q_i \right) - \sum_{j=m+1}^{n} q_j \ln \left(\sum_{i=m+1}^{n} q_i \right)$$

Again, since $\sum_{j=1}^{n} q_j \ln q_j$ is a constant, we can reduce the problem to

$$\max \; Q_m' = - \sum_{j=1}^{m} q_j \ln \left(\sum_{i=1}^{m} q_i \right) - \sum_{j=m+1}^{n} q_j \ln \left(\sum_{i=m+1}^{n} q_i \right) \qquad (1)$$

s.t. $m = 1,2,\ldots,n-1$

Let $P_m = P(1 \leq i^* \leq m) = \sum_{j=1}^{m} q_j$. Expression (1) is of the form:

$- P_m \ln P_m - (1 - P_m) \ln(1 - P_m)$. Further, P_m is an increasing function of m.

Consider the function

$f(x) = -x \ln x - (1 - x)\ln(1 - x), \quad 0 < x < 1.$

Let us maximize $f(x)$ for $0 < x < 1$.

$$f'(x) = -\ln x - 1 - [-\ln (1-x) - 1] = 0$$

$$= -\ln x + \ln (1-x) = 0$$

$$= \ln \left(\frac{1-x}{x}\right) = 0 \Rightarrow x = 1-x \Rightarrow x = 1/2$$

Further, $f''(x) = - \frac{1}{x} - \frac{1}{1-x} < 0$, for $0 < x < 1$.

Thus $f(x)$ is concave over $(0,1)$ with a single maximum at $x = 1/2$.

From the above concavity argument and the fact that $f(x)$ is symmetric about $x = 1/2$, it follows that in order to maximize Q_m', and consequently to maximize the expected reduction in uncertainty, one must choose m such that $\left| \sum_{i=1}^{m} q_i - 1/2 \right|$ is a minimum.

The problem may be formulated in another way which leads to the same result. One may consider the amount of information given by the realization of a finite scheme to be equal to the entropy of the scheme [14]. Any test defines a finite scheme with two events, A_1 = {the test fails} and its complement $\overline{A}_1$ = {the test passes}. The information gained from such a test is then

$$U = -P(A_1)\ln P(A_1) - P(\overline{A}_1)\ln P(\overline{A}_1).$$

But $P(\overline{A}_1) = 1 - P(A_1)$, so

$$U = - P(A_1)\ln P(A_1) - (1 - P(A_1))\ln (1 - P(A_1)).$$

As developed above, $P(A_1) = \sum_{i=1}^{m} q_i$ when testing across the first m components. Maximization of U subject to $m=1,2,\ldots,n-1$ is equivalent to expression (1), which now has the interpretation of maximizing the information gained by this test and hence leads to the same result as above.

B. APPLICATION TO SUCCEEDING TESTS

The above derivation applies for the first test. In all subsequent tests, the q-values for the components must be modified to reflect the additional knowledge obtained on the system state. Specifically, we want q_k^r, the probability that component k is failed given the system is in state $S(i,j)$, for $i \leq k \leq j$. Note that

$$q_k^r = \frac{q_k}{P[S(i,j)]} = \frac{q_k}{\sum_{h=i}^{j} q_h}, \quad i \leq k \leq j,$$

which is a normalization of q_k such that $\sum_{k=i}^{j} q_k^r = 1$.

This yields a "system" consisting of components i through j known to contain the defective component. Now, in order to maximize the expected reduction in uncertainty, choose m such that $\left| \sum_{h=i}^{m} q_h^r - 1/2 \right|$ is a minimum.

C. ALGORITHM FOR COMPUTATION

The following steps constitute the calculations necessary for test plan specification:

1. Calculate component posterior probabilities of failure, q_i for $i=1,2,\ldots,n$, using the algorithm of Appendix A.

2. State $S(1,n)$. Form the partial sums

$$Z(k) = \sum_{i=1}^{k} q_i$$

sequentially for $k=1,2,\ldots,$ until $Z(k) > 0.5$ for the first time. Let this $k = k_o$. Now $Z(k_o-1) < 1/2 \leq Z(k_o)$, and $d^*(1,n) = k_o-1$ or k_o, according to the following:

$$d^*(1,n) = \begin{cases} k_o-1, & \text{if } |Z(k_o-1) - 1/2| \leq |Z(k_o) - 1/2| \\ k_o, & \text{if } |Z(k_o-1) - 1/2| > |Z(k_o) - 1/2| \end{cases}$$

The first of these conditions becomes

$$1/2 - Z(k_o-1) \leq Z(k_o) - 1/2$$
$$Z(k_o) + Z(k_o-1) \geq 1$$

and the second condition becomes

$$1/2 - Z(k_o-1) > Z(k_o) - 1/2$$
$$Z(k_o) + Z(k_o-1) < 1$$

so, choose $d^*(1,n)$ according to the following:

$$d^*(1,n) = \begin{cases} k_o-1, & \text{if } Z(k_o) + Z(k_o-1) \geq 1 \\ k_o, & \text{if } Z(k_o) + Z(k_o-1) < 1 \end{cases}$$

3. Resulting state determination. If at test t, $d^*(i,j) = k$, then the two possible resulting states at test $t+1$ are $S(i,k)$ and $S(k+1,j)$.

4. State $S(i,j)$. Form the partial sums

$$Z(k) = \sum_{h=i}^{k} q_h^r$$

sequentially for $k=i,i+1,\ldots,$ until $Z(k) \geq 0.5$ for the first time. Let this $k = k_o$. As shown above, choose $d^*(i,j)$ according to the following:

$$d^*(i,j) = \begin{cases} k_o-1, & \text{if } Z(k_o) + Z(k_o-1) \geq 1 \\ k_o, & \text{if } Z(k_o) + Z(k_o-1) < 1 \end{cases}$$

5. Located failed components. Any test plan must include test sequences to locate each component, should that component be the defective. If $d^*(i,j) = i$, component i is located. If $d^*(i,j) = j-1$, component j is located. Note that $d^*(i,i+1) = i$ always and locates both components i and $i+1$.

6. Continue steps three through five until all components are located. This completes the test plan specification. Brulé, Johnson and Kletsky[3] have shown that exactly $n-1$ tests must be specified to complete the test plan.

7. Expected number of tests required. If a test plan requires a maximum of t_o tests, then

$$E(N) = \sum_{i=1}^{t_o} i \cdot P(N=i),$$

where $P(N=i)$ is the sum of the component q-values for all components requiring exactly i tests.

8. Variance of number of tests required.

$$V(N) = E(N^2) - [E(N)]^2$$

$$= \sum_{i=1}^{t_o} i^2 \cdot P(N=i) - [E(N)]^2$$

APPENDIX C

PROPERTIES OF THE SEQUENTIAL HALVING PROCEDURE

A. MAXIMUM NUMBER OF TESTS REQUIRED

Let m_t = the cardinality of the set known to contain the failed component (the defective set) after t tests have been conducted. Express m_t in the form $m_t = 2^{k_t} + r_t$, where $k_t \in \{0,1,\ldots\}$ and the remainder term, $r_t \in \{0,1,\ldots,(2^{k_t} - 1)\}$. Initially, there are $n = m_o = 2^{k_o} + r_o$ components to be tested.

Let m_t^* be the maximum number of components which could be in the defective set after the t^{th} test. The t^{th} test partitions the m_{t-1} set into two subsets containing m_t' and m_t'' components respectively, such that $m_t' + m_t'' = m_{t-1}$ and $m_t^* = \max(m_t', m_t'')$.

Let $N(\Pi)$ be the maximum number of tests which may be required to locate the failed component under any sequential testing procedure Π.

Let d_t be the number of components to be tested across on the t^{th} test.

Define the sequential halving procedure Π^* as follows:
If $m_t = 2^{k_t} + r_t$, then divide in the middle so that $d_{t+1} = 2^{(k_t-1)} + [r_t/2]$, where $[r_t/2]$ is the greatest integer less than or equal to $r_t/2$.

Lemma 1. For a system of $n = m_o = 2^{k_o} + r_o$ components, the sequential halving procedure Π^* yields

$$N(\Pi^*) = \begin{cases} k_o & , \text{ for } r_o = 0 \\ k_o + 1 & , \text{ for } r_o = 1,2,\ldots,(2^{k_o} - 1) \end{cases}$$

Proof. Case I ($r_o = 0$): $m_o = 2^{k_o}$. Under Π^*, $d_1 = 2^{k_o-1}$ and $m_1 = 2^{k_o-1}$. For the second test $d_2 = 2^{k_o-2}$ and $m_2 = 2^{k_o-2}$. For the third test $d_3 = 2^{k_o-3}$ and $m_3 = 2^{k_o-3}$. Continue this procedure until $m_{k_o-1} = 2^{k_o-(k_o-1)} = 2$. Then $d_{k_o} = 2^0 = 1$ and $m_{k_o} = 1$. Testing is now complete, since the one component remaining must in fact be faulty. Therefore, k_o tests are required and $N(\Pi^*) = k_o$.

Case II ($r_o \neq 0$): $n = m_o = 2^{k_o} + r_o$, where now $r_o \in \{1,2,\ldots,(2^{k_o} - 1)\}$. Under Π^*, $d_1 = 2^{k_o-1} + [r_o/2]$, $[r_o/2] \in \{0,1,\ldots,(2^{k_o-1} - 1)\}$. This test may divide the components into two groups with $2^{k_o-1} + [r_o/2]$ components in one group and $2^{k_o-1} + [r_o/2] + 1$ components in the other if r_o is odd, or two groups both with $2^{k_o-1} + r_o/2$ components if r_o is even. In the worse case,

$$m_1^* = 2^{k_o-1} + r_o/2 + 1$$
$$= 2^{k_o-1} + r_1 , \quad r_1 \in \{1,2,\ldots,(2^{k_o-1})\}.$$

Suppose $r_1 = 2^{k_o-1}$. Then $m_1^* = 2^{k_o-1} + 2^{k_o-1} = 2^{k_o}$. Under case I, it is shown that k_o further tests are required under Π^*, yielding $N(\Pi^*) = k_o + 1$.

Suppose alternatively that $r_1 \in \{1,2,\ldots,(2^{k_o-1}-1)\}$. Then $m_1^* = 2^{k_o-1} + r_1$, and under Π^*, $d_2 = 2^{k_o-2} + [r_1/2]$, $[r_1/2] \in \{0,1,\ldots(2^{k_o-2}-1)\}$ and

$$m_2^* = 2^{k_o-2} + [r_1/2] + 1$$
$$= 2^{k_o-2} + r_2 , \quad r_2 \in \{1,2,\ldots,(2^{k_o-2})\}.$$

Suppose $r_2 = 2^{k_o-2}$. Then $m_2^* = 2^{k_o-2} + 2^{k_o-2} = 2^{k_o-1}$. Under case I it is shown that k_o-1 further tests are required under Π^*, yielding $N(\Pi^*) = k_o + 1$.

Suppose alternatively that $r_2 \in \{1,2,\ldots,(2^{k_o-2}-1)\}$. Then $m_2^* = 2^{k_o-2} + r_2$, and under Π^*, $d_3 = 2^{k_o-3} + [r_2/2]$, $[r_2/2] \in \{0,1,\ldots,(2^{k_o-3}-1)\}$ and

$$m_3^* = 2^{k_o-3} + [r_2/2] + 1$$

$$= 2^{k_o-3} + r_3 \ , \quad r_3 \quad 1,2,\ldots,(2^{k_o-3}) \ .$$

Continue this procedure until

$$m_{k_o}^* = 2^{k_o-k_o} + [r_{k_o-1}/2] + 1$$

$$= 2^0 + r_{k_o} \ , \quad r_{k_o} \in \{1,2,\ldots,(2^{k_o-k_o})\}$$

$$= 1 + 1 = 2$$

Then $d_{k_o+1} = 1$ and $m_{k_o+1}^* = 1$. As in case I above, testing is now complete and $N(\Pi^*) = k_o + 1$. Q. E. D.

B. MINIMAX PROPERTY

Theorem 1. Of all dichotomous sequential testing procedures Π, the sequential halving procedure Π^* is a minimax procedure. That is,

$$N(\Pi^*) = \min_{\Pi} N(\Pi)$$

Proof. The theorem is true vacuously for $n = 1$, since no testing is required. It is also true for $n = 2$ and $n = 3$, since Π^* is the only dichotomous sequential testing procedure to follow. Note here that for $n = 3$, $d_1 = 2$ (testing across the first two components) is equivalent to testing across the third component which satisfies Π^*.

Assume the theorem to be true for $n = 1,2,\ldots,(s-1)$. To show: The theorem holds for $n = s$.

Let $n = s = 2^{k_o} + r_o$, $r_o \in \{0,1,\ldots,(2^{k_o}-1)\}$. Any procedure Π reduces at each test the number of components in the defective set by at least 1. So for any procedure Π, $m_1^* \leq s - 1$. It is assumed that the theorem holds for $n = 1,2,\ldots,(s-1)$. Thus Π^* is the minimax procedure to follow after conducting the first test. Therefore, we need consider those procedures Π' which differ from Π^* only at the first test.

Case I ($r_o = 0$): $s = m_o = 2^{k_o}$. Under any procedure Π', $2^{k_o-1}+2 \leq m_1^* \leq 2^{k_o}-1$. Employing Π^* for subsequent tests yields (by Lemma 1): $N(\Pi') = k_o+1 > k_o = N(\Pi^*)$.

Case II ($r_o \neq 0$): $s = m_o = 2^{k_o} + r_o$, $r_o \in \{1,2,\ldots(2^{k_o}-1)\}$. Under any procedure Π',

$$2^{k_o-1} + [r_o/2] + 2 < m_1^* \leq 2^{k_o} + r_o - 1,$$

where $r_o \in \{1,2,\ldots(2^{k_o}-1)\} \Rightarrow [r_o/2] \in \{0,1,\ldots,(2^{k_o-1}-1)\}$. So, $2^{k_o-1} + [r_o/2] + 2 \geq 2^{k_o-1} + 2$, for all r_o, and $2^{k_o} + r_o - 1 \geq 2^{k_o}$, for all r_o. Thus, for $r_o \neq 0$ there exists no procedure Π' for which $m_1^* < 2^{k_o-1} + 2$. Therefore, employing Π^* on subsequent tests yields $N(\Pi') = k_o+1 = N(\Pi^*)$ for those procedures for which $m_1^* \leq 2^{k_o}$, and $N(\Pi') = k_o+2 > k_o+1 = N(\Pi^*)$ for those procedures for which $m_1^* > 2^{k_o}$. Q. E. D.

C. COUNTEREXAMPLE TO MINIMAX PROPERTIES OF "HALF-SPLIT TECHNIQUE"

As cited in Section II, Brulé, Johnson and Kletsky [3] make two claims involving minimax properties of the "half-split technique". They claim that it minimizes the maximum possible cost in both the equal cost-equal probability and the equal cost-unequal probability cases.

Since the costs are equal for all tests in both cases, we may let the cost of a test be unity. Then the maximum possible cost of a test plan is simply the maximum number of tests which may be required.

Let the system consist of 15 components. The "half-split technique" prescribes that if $m_t = 2^{k_t} + r_t$ components remain in the defective set after t tests, $t = 0,1,2,\ldots,$ then the $(t+1)^{st}$ test must partition these components into two groups such that each group contains at least 2^{k_t-1} components. One may generate the test plan shown in Figure 21, which follows this procedure. Under this plan the maximum possible cost is 6. From Lemma 1 above, we know that employing the sequential halving procedure will lead to a maximum possible cost of 4. Thus the above solution is not minimax. Note also that a minimax criterion is independent of component reliabilities.

Test	State	Dec	Comp
1	1,15	4	
2	1, 4	2	
	5,15	8	
3	1, 2	1	1, 2
	3, 4	3	3, 4
	5, 8	5	5
	9,15	10	
4	6, 8	6	6
	9,10	9	9,10
	11,15	12	
5	7, 8	7	7, 8
	11,12	11	11,12
	13,15	13	13
6	14,15	14	14,15

"Half-Split Technique" Test Plan

Figure 21

COMPUTER OUTPUT

COMPONENT	P(I)	Q(I)
1	0.4050	0.1330
2	0.4350	0.1176
3	0.4650	0.1042
4	0.4950	0.0924
5	0.5250	0.0819
6	0.5550	0.0726
7	0.5850	0.0642
8	0.6150	0.0567
9	0.6450	0.0498
10	0.6750	0.0436
11	0.7050	0.0379
12	0.7350	0.0326
13	0.7650	0.0278
14	0.7950	0.0233
15	0.8250	0.0192
16	0.8550	0.0154
17	0.8850	0.0118
18	0.9150	0.0084
19	0.9450	0.0053
20	0.9750	0.0023

SYSTEM STATE	OPTIMAL DECISION	MIN EXPECTED NR TESTS	LOCATED FAILED COMPONENTS
TEST 1			
(1, 20)	5	3.9152	
TEST 2			
(1, 5)	2	2.3294	
(6, 20)	9	3.5732	
TEST 3			
(1, 2)	1	1.0000	1 2
(3, 5)	3	1.6259	3
(6, 9)	7	2.0000	
(10, 20)	12	3.1860	
TEST 4			
(4, 5)	4	1.0000	4 5
(6, 7)	6	1.0000	6 7
(8, 9)	8	1.0000	8 9
(10, 12)	10	1.6180	10
(13, 20)	14	2.7571	
TEST 5			
(11, 12)	11	1.0000	11 12
(13, 14)	13	1.0000	13 14
(15, 20)	16	2.3785	
TEST 6			
(15, 16)	15	1.0000	15 16
(17, 20)	17	1.8497	17
TEST 7			
(18, 20)	18	1.4744	18
TEST 8			
(19, 20)	19	1.0000	19 20
VARIANCE OF NR OF TESTS =		0.8517	

```
C
C     DYNAMIC PROGRAMMING FORMULATION OF FAULT LOCATION PROBLEM
C
C     OBJECTIVE.  GENERATE TEST PLAN FOR AN N-COMPONENT SYSTEM
C                 WHICH MINIMIZES THE EXPECTED NUMBER OF TESTS
C                 REQUIRED TO LOCATE THE SINGLE FAILED COMPONENT.
C
C     INPUTS.  N = THE NUMBER OF COMPONENTS WHICH COMPRISE THE SYSTEM.
C              MAX = ALLOWABLE N IS DIMENSION OF P VECTOR IN COMMON
C                    STATEMENTS.
C              P(I) = THE A PRIORI RELIABILITY OF COMPONENT I.  EACH
C                     P(I) MUST BE STRICTLY GREATER THAN ZERO.
C
C     MAIN PROGRAM.  READS IN INPUTS.
C                    CALCULATES COMPONENT POSTERIOR PROBABILITIES OF
C                         FAILURE GIVEN SYSTEM FAILED WITH EXACTLY ONE
C                         FAILED COMPONENT, Q(I).
C                    PRINTS P(I) AND Q(I) VALUES FOR ALL COMPONENTS.
C
      DOUBLE PRECISION FM(40,40)
      COMMON ID(40), P(40), Q(40), FM, NBCX(40)
      READ (5,10) N
   10 FORMAT (I5)
      READ (5,15) (P(I), I = 1,N)
   15 FORMAT (7F10.4)
      SUM = 0.0
      DO 16 I = 1,N
      SUM = SUM + (1.0 - P(I))/P(I)
   16 CONTINUE
      DO 17 I = 1,N
      Q(I) = ((1.0 - P(I))/P(I))/SUM
   17 CONTINUE
      WRITE (6,18)
   18 FORMAT (1H1,////////16X, 'COMPONENT', 10X, 'P(I)', 16X, 'Q(I)'//)
      DO 19 I = 1,N
      WRITE (6,20) I, P(I), Q(I)
   19 CONTINUE
   20 FORMAT (1H ,17X, I3, 13X, F6.4, 14X, F6.4)
      CALL DPROG (N)
      STOP
      END
C
C
C
```

```
      SUBROUTINE DPROG (N)
C
C       CALCULATES TEST PLAN WHICH MINIMIZES EXPECTED NUMBER OF TESTS
C         REQUIRED.
C       CALCULATES EXPECTED NUMBER OF TESTS REQUIRED.  THIS VALUE
C         APPEARS IN OUTPUT UNDER 'MIN EXPECTED NR TESTS' COLUMN
C         ADJACENT TO STATE S(1,N).
C       PRINTS TABULAR FORM OF TEST PLAN SPECIFICATION.
C
C
C       ENTITIES.  FM(I,J) = MATRIX CONTAINING MINIMUM EXP NR TESTS
C                            BEGINNING AT STATE S(I,J), AND OPTIMUM
C                            DECISION VARIABLE AT THAT STATE.
C                  ID = INDICATOR VECTOR WHOSE COMPONENTS TAKE ON
C                       THE VALUES 0, 1, OR 2.  ZERO INDICATES THE
C                       COMPONENT HAS NOT BEEN THE RIGHT END COM-
C                       PONENT FOR A TEST, ONE INDICATES THE COM-
C                       PONENT HAS BEEN THE RIGHT END COMPONENT
C                       FOR A TEST.  TWO INDICATES THE COMPONENT
C                       HAS BEEN LOCATED.
C                  NBOX = VECTOR CONTAINING THE NUMBER OF TESTS
C                         REQUIRED TO LOCATE EACH COMPONENT.  USED IN
C                         SUBROUTINE VAR TO CALCULATE VARIANCE OF NR
C                         OF TESTS REQUIRED.
C                  QVAL  CORRESPONDS TO  Q  FUNCTION IN D.P. FORMU-
C                        LATION.  ITS MINIMUM VALUE FOR ANY STATE IS
C                        STORED IN QBEST TO BE ENTERED INTO FM ARRAY.
C                  THOUS = 1000.0 IN DOUBLE PRECISION.
C
      DOUBLE PRECISION FM(40,40)
      COMMON ID(40), P(40), Q(40), FM, NBOX(40)
      DOUBLE PRECISION QVAL, QBEST, THOUS
      WRITE (6,30)
   30 FORMAT (1H1//////////30X, 'OPTIMAL', 5X, 'MIN EXPECTED', 3X,
     1   'LOCATED FAILED'/14X, 'SYSTEM STATE', 4X, 'DECISION', 7X,
     2   'NR TESTS', 7X, 'COMPONENTS'//)
      DO 40 I = 1,N
      DO 40 J = 1,N
      FM(I,J) = 0.0
   40 CONTINUE
C
C       ENTER INITIAL VALUES IN FM ARRAY.
C
      NLESS = N - 1
      DO 50 I = 1,NLESS
      FM(I,I+1) = I*1000 + 1.0
      FM(I,I) = 1000.0
   50 CONTINUE
```

```
      FM(N,N) = 1000.0
C
CC      BEGIN LOOP TO STEP INTERVAL WIDTH.
C
      DO 400 INT = 2,NLESS
      LMAX = N - INT
C
CC      BEGIN LOOP TO STEP INTERVAL POSITION.
C
      DO 300 L = 1,LMAX
      M = L + INT
C
CC      CALCULATE PROBABILITY COEFFICIENT (PCOEFF).
CC      PCOEFF = PROBABILITY STATE S(L,K) RESULTS BEGINNING AT STATE
CCC             S(L,M) AND TESTING AT THE NEXT TEST ACROSS THE FIRST
CCC             K  COMPONENTS.
C
      PDENOM = 0.0
      DO 100 I = L,M
      PDENOM = PDENOM + Q(I)
  100 CONTINUE
      PNUM = 0.0
      QBEST = 9999
      MLESS = M - 1
C
CC      BEGIN LOOP TO STEP DECISION VARIABLE, K.
C
      DO 200 K = L,MLESS
      PNUM = PNUM + Q(K)
      PCOEF = PNUM/PDENOM
C       CALCULATE Q FUNCTION FOR THIS COMBINATION OF L, M AND K.
      THOUS = 1000.0
      QVAL = PCOEF * (1.0 + DMOD(FM(L,K),THOUS)) + (1.0 - PCOEF) *
     1  (1.0 + DMOD(FM(K+1,M),THOUS))
C       STORE MIN Q FUNCTION VALUE IN QBEST AND CORRES. K IN KBEST.
      IF (QVAL.GE.QBEST) GO TO 200
      QBEST = QVAL
      KBEST = K
  200 CONTINUE
C       STORE MIN Q AND CORRES. K IN F MATRIX.
      FM(L,M) = (KBEST * 1000.0) + QBEST
  300 CONTINUE
  400 CONTINUE
      DO 410 I = 1,NLESS
      IO(I) = 0
  410 CONTINUE
C
C       ISTAGE = I  INDICATES THAT TEST  I  IS BEING CONDUCTED.
```

```
C       NFAIL  INDICATES HOW MANY COMPONENTS HAVE BEEN LOCATED.
C
      ISTAGE = 1
      NFAIL = C
      ID(N) = 1
      WRITE (6,500) ISTAGE
  500 FORMAT (1HC, 15X, 'TEST', I5)
      KD = IFIX(SNGL(FM(1,N)/1000.0))
      IL = 1
      F = FM(1,N) - (KD * 1000.0)
C
C       EXPVAL IS THE MINIMUM NR OF TESTS REQUIRED.
C
      EXPVAL = F
C
C       CHECK FOR LOCATED FAILED COMPONENTS ON THE FIRST TEST.
C
      IF (KD.EQ.1) GO TO 510
      IF (KD.EQ.N-1) GO TO 520
      WRITE (6,700) IL, N, KD, F
      GO TO 540
  510 WRITE (6,720) IL, N, KD, F, IL
      NBOX(IL) = ISTAGE
      ID(KD) = 2
      NFAIL = NFAIL + 1
      GO TO 810
  520 WRITE (6,720) IL, N, KD, F, N
      NBOX(N) = ISTAGE
      ID(N) = 2
      ID(N-1) = 1
      NFAIL = NFAIL + 1
      GO TO 810
  540 ID(KD) = 1
      GO TO 810
  550 IL = 1
  600 DO 800 I = IL,N
      IF (ID(I)-1) 800,605,790
  605 KD = IFIX(SNGL(FM(IL,I)/1000.0))
      ID(KD) = 1
      F = FM(IL,I) - (KD * 1000.0)
      IF (I-IL.NE.1) GO TO 610
      NFAIL = NFAIL + 2
      ID(IL) = 2
      ID(I) = 2
      WRITE (6,710) IL, I, KD, F, IL, I
      NBOX(IL) = ISTAGE
      NBOX(I) = ISTAGE
      GO TO 790
```

```
  610 IF (I-IL.NE.2) GO TO 630
      NFAIL = NFAIL + 1
      IF (KD-IL.NE.1) GO TO 620
      ID(I) = 2
      WRITE (6,720) IL, I, KD, F, I
      NBOX(I) = ISTAGE
      GO TO 790
  620 ID(IL) = 2
      WRITE (6,720) IL, I, KD, F, IL
      NBOX(IL) = ISTAGE
      GO TO 790
  630 IF (KD.EQ.IL) GO TO 640
      IF (ID(KD+1).EQ.1) GO TO 650
      WRITE (6,700) IL, I, KD, F
      GO TO 790
  640 NFAIL = NFAIL + 1
      GO TO 620
  650 NFAIL = NFAIL + 1
      ID(KD+1) = 2
      KDPLUS = KD + 1
      WRITE (6,720) IL, I, KD, F, KDPLUS
      NBOX(KDPLUS) = ISTAGE
      GO TO 790
  700 FORMAT (1H , 15X, '(', I3, ',', I3, ')', 6X, I3, 9X, F8.4)
  710 FORMAT (1H , 15X, '(', I3, ',', I3, ')', 6X, I3, 9X, F8.4,
     1   10X, I3, I4)
  720 FORMAT (1H , 15X, '(', I3, ',', I3, ')', 6X, I3, 9X, F8.4,
     1   10X, I3)
  790 IF (I.EQ.N) GO TO 800
      II = I
      GO TO 900
  800 CONTINUE
      IF (NFAIL.EQ.N) GO TO 9999
  810 ISTAGE = ISTAGE + 1
      IF (ISTAGE.GE.N) RETURN
      WRITE (6,500) ISTAGE
      GO TO 550
  900 IF (NFAIL.EQ.N) GO TO 9999
      IL = II + 1
      GO TO 600
 9999 CALL VAR (N, EXPVAL)
      RETURN
      END
C
C
C
```

```
      SUBROUTINE VAR (N, EXPVAL)
      DOUBLE PRECISION FM(40,40)
      COMMON ID(40), P(40), Q(40), FM, NBOX(40)
      SUM = 0.0
      DO 1000 I = 1,N
      SUM = SUM + NBOX(I)*NBOX(I)*Q(I)
1000  CONTINUE
      VARN = SUM - EXPVAL*EXPVAL
      WRITE (6,1100) VARN
1100  FORMAT (1HO, 15X, 'VARIANCE OF NR OF TESTS = ', F9.4)
      RETURN
      END
```

BIBLIOGRAPHY

1. Black, William, "Discrete Sequential Search," Information and Control, v. 8, p. 159-162, 1965.

2. Blackwell, D., Notes on Dynamic Programming, Department of Statistics, University of California, class notes, 1962.

3. Brulé, J. D., Johnson, R. A., and Kletsky, E. J., "Diagnosis of Equipment Failures," IRE Transactions on Reliability and Quality Control, April 1960.

4. Butterworth, R., Some Reliability Fault Testing Models, to appear.

5. Columbia University Technical Report 36, A Note on Sequential Search, by Morton Klein, 30 August 1967.

6. Coordinated Science Laboratory Report R-409, Improving the Diagnosability of Modular Combinational Logic by Test Point Insertion, by T. G. Gaddess, March 1969.

7. Dorfman, R. "The Detection of Defective Members of Large Populations," Annals of Mathematical Statistics, v. 14, p. 436-440, 1943.

8. Finucan, H. M., "The Blood Testing Problem," Applied Statistics, v. 13, p. 43-50, 1964.

9. Firstman, S. I. and Gluss, B., "Optimum Search Routines for Automatic Fault Location," Operations Research, v. 8, p. 512-523, 1960.

10. Gluss, B., "An Optimum Policy for Detecting a Fault in a Complex System," Operations Research, v. 7, p. 468-477, 1959.

11. Gruman Aircraft Engineering Corporation Note ADN 07-05-68.1, Information Theory Approach to Testing, by E. Kovacs, April 1968.

12. Huffman, D. A., "A Method for the Construction of Minimum Redundancy Codes," IRE Proceedings, v. 40, p. 1908, 1952.

13. Johnson, R. A., "An Information Theory Approach to Diagnosis," Proceedings Sixth National Symposium on Reliability and Quality Control, Washington, D. C., 11-13 January 1960.

14. Khinchin, A. I., Mathematical Foundations of Information Theory, Dover Publications, Inc., 1957.

15. Kletsky, E. J., "An Application of the Information Theory Approach to Failure Diagnosis," IRE Transactions on Reliability and Quality Control, December 1960.

16. Matula, D., "A Periodic Optimal Search," American Mathematical Monthly, v. 71, p. 15-21, 1964.

17. Nemhauser, G. L., Introduction to Dynamic Programming, Wiley, 1966.

18. Rand Corporation Report RM-1652, Optimal Sequential Testing, by S. M. Johnson, March 1956.

19. Sandelius, M., "On an Optimal Search Procedure," American Mathematical Monthly, v. 68, p. 133-134, 1961.

20. Sobel, M., "Group Testing to Classify Efficiently All Units in a Binomial Sample," Information and Decision Processes, edited by R. E. Machol, McGraw-Hill, 1960.

21. Sobel, M. and Groll, P. A., "Group Testing to Eliminate Efficiently All Defectives in a Binomial Sample," Bell Systems Technical Journal, v. 38, p. 1179-1252, 1959.

22. Stanford University Research Institute Report EE577-594T1, Diagnosis of Equipment Failures, by J. D. Brulé, R. A. Johnson, and E. J. Kletsky, April 1959.

23. Stanford University Technical Report 72, Optimal Group Testing, by M. Sobel, 10 July 1964.

24. Stanford University Technical Report 131, Finding a Single Defective in Binomial Group Testing, by S. Kumar and M. Sobel, August 1970.

25. Ungar, P., "The Cutoff Point for Group Testing," Communications on Pure and Applied Mathematics, v. 13, p. 49-54, 1960.

26. Zimmerman, S., "An Optimal Search Procedure," American Mathematical Monthly, v. 66, p. 690-693, 1959.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst Professor R. W. Butterworth, Code 55 Bd Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
4.	LT David R. Campbell, USN 1053 Halsey Drive Monterey, California 93940	1
5.	Assoc Professor Donald Barr, Code 55 Bn Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
6.	TRW Systems Group One Space Park Redondo Beach, California 90278 Attn: Mr. A. Dobieski	1
7.	Chief of Naval Personnel Pers-11b Department of the Navy Washington, D. C. 20370	1
8.	Naval Postgraduate School Department of OR and AS Monterey, California 93940	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

An Information Theory Approach to a Fault Location Problem

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; September 1971

5. AUTHOR(S) (First name, middle initial, last name)

David Russell Campbell

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1971	82	26

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

The fault location model under investigation consists of an n-component series system known to have exactly one failed component. Component positions in the system are taken as fixed. A component is either working or failed. Components work or fail independently of each other, with their a priori reliabilities taken as given but not necessarily equal. Group testing to locate the failed component is sequential, binary and dichotomous in nature with certain results. The only costs are the number of tests made. The three solution procedures investigated are (1) a dynamic programming formulation, (2) a sequential halving procedure, and (3) a procedure based on information theory. The criteria for optimality are minimization of the expected number of tests required and minimization of the maximum number of tests required.

D FORM 1 NOV 65 1473 (PAGE 1)

0101-807-6811

Security Classification

A-31408

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Fault Location Model						
Series System						
Group Testing						
Information Theory						
Dynamic Programming						

Thesis 131515
.C1933 Campbell
c.1 An information theory approach to a fault location problem.